TRABAJO FINAL PARA OBTENER EL GRADO DE LICENCIADA EN INFORMÁTICA

Arquitectura para aplicaciones Web personalizadas

Autora: Mercerat Bárbara

Facultad de Informática
Universidad Nacional de La Plata

Director: Dr. Gustavo H. Rossi

La Plata, Noviembre de 2005

Autora: Mercerat Bárbara

Introducción **3**
Conceptos de Personalización **7**
1.1 Motivación 8
1.2 ¿Qué es la Personalización? 9
1.3 Características de la Personalización 10
1.4 Personalización en la Web 13
1.4.1. Características que pueden Personalizarse 15
1.4.2. Customización y Personalización 17
1.4.3. Personalización y Privacidad 17
Características de las Aplicaciones Personalizadas **19**
2.1 Introducción 20
2.2 Dinamismo de la Información 21
2.3 Identificación de los Usuarios 22
2.4 Perfil de Usuario 25
2.4.1. ¿Qué representa un Perfil de Usuario? 26
2.4.2. Información de los Usuarios 26
2.4.3. Obtención de la Información de Usuario 28
2.5 Reglas de Negocio 31
2.5.1. ¿Qué representan las Reglas de Negocio? 31
2.5.2. Aplicaciones de las Reglas de Negocio a la Personalización 32
Trabajos Relacionados **33**
3.1 Motivación 34
3.2 Mediator que comunica Usuarios con Servicios Personalizados 34
3.3 Utilización de Smart Cards como Almacenamiento 36
3.4 Utilización de un Proxy como Almacenamiento 38
3.5 Recursos compartidos entre Usuarios con Intereses Similares 40
Metodología y Patrones Utilizados **43**
4.1 Metodologías examinadas 44
4.2 Metodología OOHDM 45
4.2.1. Diseño Conceptual 45
4.2.2. Diseño Navegacional 46
4.2.3. Diseño de Interfaz Abstracta 46
4.2.4. Implementación 47
4.3 Utilización del Patrón MVC 47
4.4 Patrones de Personalización 49
4.4.1. Link Personalizado 49
4.4.2. Contenido Personalizado 51
4.4.3. Estructura Personalizada 51
4.4.4. Client-side Personalizado 53
4.5 Aportes de la Metodología y Patrones a la creación de la Arquitectura 55
Diseño General de la Arquitectura **57**
5.1 ¿Cómo será presentada la Arquitectura? 58
5.2 Características generales 58
5.3 Utilizando el Patrón MVC en la Arquitectura 59
5.3.1. Flujo de la Información 61
5.3.2. Diferentes tecnologías utilizadas en la interfaz 63
5.4 Implementación del MVC 64
5.5 Conclusiones 67
Modelo Interno de la Arquitectura **68**
6.1 Diseño Interno del Modelo 69
6.1.1. ¿Cómo surgen los módulos? 70
6.2 Perfil de Usuario 71
6.2.1. Ubicación del Perfil (servidor vs. cliente) 73
6.3 Módulos que componen el Modelo 74

6.3.1. Modelo Navegacional ... 76
6.3.2. Modelo de Aplicación ... 77
6.3.3. Reglas de Negocio ... 78
6.3.4. Modelo de Interfaz ... 79
6.4 Implementación de los Modelos y el Perfil de Usuario ... 81
6.5 Conclusión ... 84
Funcionamiento de la Arquitectura ... 86
7.1 Creación de los Nodos ... 87
7.2 Facilidad Agregado y Eliminación de Aspectos personalizables ... 89
Conclusiones ... 93
Tabla de Figuras ... 95
Referencias ... 96

Introducción

Desde los años 60, que fue creada, hasta el día de hoy, Internet ha sufrido un cambio de abrupto crecimiento. Lo que en sus principios fue solo un medio de comunicación empleado por el ejército de los EEUU, con el tiempo fue creciendo y abriéndose al mundo, y cualquier persona con fines académicos o de investigación podía tener acceso a la red. El propósito de este medio fue cambiando con la creación del primer explorador Web, dando inicio a la World Wide Web. Lo que produjo una auténtica revolución originando el comienzo de la creación de sitios Web. Al principio, con fines de investigación e intercambio de información, pero luego, fueron utilizados con fines de negocio. Con este cambio de enfoque, Internet logró un empuje importantísimo en su desarrollo, lo que produjo en los últimos años un enorme crecimiento de la World Wide Web, dejando de ser utilizada solo por cierta gente, y convirtiéndose en un medio de comunicación masiva, pudiendo ser accedida por cualquier persona alrededor del mundo. Por este motivo, este medio de comunicación se convirtió en uno de los más empleados de los últimos tiempos.

La World Wide Web, en sus principios destinada solamente a cierto tipo de información, hoy en día es utilizada para la venta de productos, marketing de empresas, búsqueda de información, contratación de servicios, entre otro millón de utilidades obtenidas a través de este medio. Esta nueva perspectiva y la ventaja de la difusión masiva, ocasionó que se acentuara la competencia de los sitios Web para atraer a los usuarios a sus sitios. Con motivo de esta competencia comenzó la evolución de las tecnologías que se utilizan para crear, diseñar, desarrollar e implementar sitios Web, para mejorar día a día los sitios en aspecto visual, rapidez y eficacia de búsqueda. Al mismo tiempo, continúan las búsquedas de novedades para atraer a los usuarios.

Una de las ideas más revolucionarias de los últimos años, fue la creación de aplicaciones Web personalizadas, sitios que brindan a cada usuario particular un ambiente más amigable, facilitando la búsqueda de información e interacción con el mismo. En otras palabras, sitios que se adaptan a cada usuario de una manera personal. Estas aplicaciones son mucho más complejas de realizar que las aplicaciones que no contemplan el aspecto de personalización. Esta complejidad se produce al intentar obtener la adaptabilidad a cada uno de los usuarios particulares. Se debe alcanzar realizar un único modelo que contenga las características generales de la aplicación y se adapte a cada uno de los usuarios según sus propiedades y preferencias. Para lograr este propósito, las aplicaciones personalizadas deben contar con información de cada usuario particular y poseer la lógica para manipular dicha información. Al añadir estas responsabilidades a la aplicación, surgen dificultades en el diseño e implementación. Lograr un buen diseño de la aplicación, pensando en flexibilidad y adaptación a toda la complejidad que significa adicionar personalización, tales como manejo de la información de cada uno de los usuarios, algoritmos de inferencia para realizar recomendaciones, adaptaciones al usuario, reglas de negocio para los usuarios, y demás, añade un grado de dificultad muchísimo mayor al que se presenta en las aplicaciones que no poseen personalización.

En este trabajo se presenta una arquitectura modular que expone una solución a las aplicaciones Web personalizadas, enfrentando la construcción de la aplicación en módulos para facilitar el desarrollo de la misma. El objetivo de la separación en módulos es la distribución de responsabilidades, proveyendo cada uno de ellos responsabilidades

específicas. De este modo, cada módulo, si bien están muy vinculados entre sí, puede ser modificado y diseñado de manera independiente uno de los otros, entre otras ventajas.

¿Por qué una Arquitectura para Aplicaciones Web de Comercio Electrónico?

Cuando se hace referencia a aplicaciones Web personalizadas se abarca un sin fin de posibilidades que se presentan en el ámbito Web. Dichas aplicaciones pueden ir desde sitios de comercio electrónico, sitios de empresas, sitios científicos, sitios informativos, sitios de foros, y un millón más de posibilidades que se pueden encontrar. Cada uno de ellos personaliza con el propósito de que el usuario se sienta cómodo e interesado en la información que le presenta el sitio, en cambio, los objetivos y funcionalidades de dichas aplicaciones son muy diferentes en cada uno de los casos. Por ejemplo, la personalización en el comercio electrónico se basa en adaptar el sitio a los clientes para vender productos, ya sean libros, indumentaria, propiedades inmobiliarias entre otras. En estos casos, los usuarios son vistos como clientes del sitio Web. El sitio se personaliza para cada uno de ellos con las preferencias de los mismos para adaptarlo a sus necesidades, facilidades de uso y gustos con el objetivo de lograr vender la mayor cantidad de productos. Para lograr esto se debe recaudar la mayor cantidad de información posible sobre los usuarios para aplicar las estrategias de negocio con el objetivo de atraerlos y mantenerlos. Esta información, es recaudada basándose en tres aspectos fundamentales, en la información propia del usuario, el comportamiento del usuario y los productos de su interés. La información propia del usuario, serían los datos personales de los mismos, nombre, edad, nacionalidad, profesión, y demás. Y la información que se guarda del usuario que no son datos personales pero son de interés para un sitio de comercio electrónico como son, los datos de historial del usuario, la frecuencia de visita al sitio, el tiempo de permanencia, los tipos de productos en los que se interesa, el tipo de pago, la frecuencia de compra, la cantidad de compras en períodos de tiempos preestablecidos, y muchas otras variables importantes para resolución de la personalización de la aplicación. Por otro lado, por ejemplo, en los sitios de manejo interno de empresas el objetivo es mantener y manejar los productos que dicha empresa produce o da como servicios. En este tipo de aplicaciones personalizadas, existen los diferentes roles que cumplen los empleados de la misma. Estos roles pueden ser, en una empresa de producción de electrodomésticos, el rol de gerente, de técnico reparador, de facturación, el rol de atención al cliente, entre otros. En estos casos, se tiene bastante información sobre el comportamiento de los usuarios de ante mano, solo basándose en el rol que cumple el mismo dentro de la empresa. La diferencia principal que se puede observar entre estos dos tipos de aplicación es a qué usuarios están orientadas las aplicaciones. En el sitio de comercio electrónico las aplicaciones son orientadas a usuarios nuevos y clientes de la tienda de comercio electrónico, en cambio, en la aplicación de las empresas los usuarios son más estáticos y no tienen elección de sitios, por este motivo, la personalización es más acotada. En estos casos, no se debe inferir tanta información y es más simple de realizar que un sitio de comercio electrónico en donde se debe manejar mucha más cantidad de información de los usuarios, y la misma es muy variada y dinámica, para lograr atraer y mantener interesados a sus usuarios.

Este trabajo se encuentra orientado a aplicaciones Web de comercio electrónico, contemplado el hecho que los sitios de comercio electrónico son más complejos y abarcan más variables dinámicas y más información sobre los usuarios, además de utilizar reglas de negocio de las cuales se hablará luego. La decisión de acotar este trabajo a este tipo de aplicaciones se basa en el sentido de que la arquitectura se centre en un problema restringido que es bastante complicado en sí mismo y no se pretenda hacer una arquitectura que abarque todo tipo de aplicación Web personalizada lo cual sería demasiado complicado y ambicioso. Esta arquitectura si bien se orienta a este tipo de

aplicación es amplia y genérica como para poder ser adaptada a otros tipos de aplicaciones Web personalizadas.

Como primera instancia de este trabajo se explican de manera detallada y completa las características y los aspectos que se presentan en las aplicaciones personalizadas. Luego de presentado el contexto general para comprender y conocer todos los aspectos que se relacionan con estas aplicaciones, se ingresa en el contenido fundamental de este trabajo, el cual se encuentra orientado a lograr soportar aplicaciones Web personalizadas de gran envergadura y con un cierto grado de complejidad

En este marco se desarrolla esta tesis que aporta una visión diferente para facilitar el desarrollo de las aplicaciones Web personalizadas, presentando una arquitectura que permite la construcción modular de aplicaciones Web personalizadas de comercio electrónico, con el objetivo de lograr un buen desarrollo y un desempeño relativamente simple. Esta arquitectura, se basa en una visión orientada al comportamiento interno de la aplicación dividiéndola en módulos. La finalidad de esta tesis es desarrollar una arquitectura pensando de una manera modular separando los módulos por funcionalidad. Con dicha división, se logra independizar el desarrollo de cada módulo provocando que las tareas de desarrollo y mantenimiento de la arquitectura puedan ser realizadas de una manera más sencilla, aportando a los diseñadores y desarrolladores de este tipo de aplicaciones independencia, por funcionalidad, entre los diferentes aspectos de la aplicación. Logrando además, que el impacto que puedan provocar las tareas de mantenimiento y posibles modificaciones que deban realizarse a las aplicaciones no sea enormes, consiguiendo efectuar estas tareas realizando cambios mínimos o posibles de contemplar y que sean llevados a cabo sin producir cambios drásticos en la aplicación.

Este trabajo está organizado en 8 capítulos. En primer lugar se presentan conceptos de la personalización, qué es la personalización y características y aspectos que pueden ser personalizados en una aplicación personalizada. Dentro de las características y aspectos se describen cuáles de los aspectos de una aplicación son tomados en cuenta para adaptar el sitio al usuario, y con respecto a las características, se presentan y describen los elementos de un sitio Web que pueden ser adaptados a los usuarios. Esto comprende el capítulo 1 de este trabajo.

El capítulo 2, no se encuentra orientado al nivel conceptual de las aplicaciones personalizadas, sino a la parte técnica y de funcionalidad de las mismas. Comprende los temas de, dinamismo de la información, recolección de la información de los usuarios, perfil de usuarios, descripción y objetivo del mismo, y reglas de negocio, que son las reglas que aparecen en las aplicaciones de comercio.

El capítulo 3 se basa en la investigación que se realizó de otras arquitecturas que soportan aplicaciones Web personalizadas, nombrando cada una de ellas y proporcionando una breve descripción de las mismas, con el objetivo de presentar otras maneras y visones de realizar las aplicaciones personalizadas.

En el siguiente capítulo, capítulo 4, se presentan y describen las metodologías y el patrón estudiados para realizar el desarrollo de la arquitectura propuesta. Posteriormente, se presentan cuatro patrones de personalización de Web que brindaron grandes aportes a la realización de la arquitectura. Se describen y proporcionan las características de las metodologías y los patrones. Por último, en la sección final, se exponen los aportes que éstos brindaron a la creación de la arquitectura.

Luego de haber proporcionado una introducción a la personalización, de otras arquitecturas existentes que soportan aplicaciones personalizadas, y de haber presentado las metodologías utilizadas en la arquitectura desarrollada que conforman los contenidos de base para la realización de este trabajo, se prosigue con los tres capítulos principales,

capítulos 5, 6 y 7. Estos capítulos presentan el núcleo central de esta tesis donde se expone la arquitectura propuesta para soportar aplicaciones Web personalizadas.

La descripción de cómo se compone la arquitectura se encuentra realizado en dos niveles. El primer nivel que la compone se presenta en el capítulo 5, el cual se organiza de la siguiente manera, en primer lugar una descripción de las características generales de la arquitectura, y luego se realiza una descripción de cada uno de los módulos que compone este nivel exponiendo su funcionalidad y la interacción que se realiza entre ellos, presentando como última sección una posible implementación.

A continuación se describe en el capítulo 6, el segundo nivel que compone la arquitectura. Este segundo nivel se halla dividido en cuatro módulos que lo integran, como primer punto se exponen las razones que dieron origen a cada uno de dichos módulos. Seguidamente, se realiza una descripción y se explica el funcionamiento y responsabilidades de cada uno de estos cuatro módulos. Una vez presentados se exhibe una implementación de ellos, exponiendo las razones de las decisiones tomadas.

En el capítulo 7 se describe el funcionamiento completo de la arquitectura. Se expresa cómo se realiza la comunicación entre los dos niveles que componen a la arquitectura, y se explica cómo se realiza la interacción entre los diferentes módulos para completar la tarea del armado del nodo personalizado en respuesta al pedido realizado por un usuario. Al término de este capítulo se expone cómo debe ser realizado el agregado o eliminación de aspectos a personalizar y se muestra qué módulos se ven afectados con dichos cambios.

Finalizando este trabajo, en el capítulo 8 se exponen las conclusiones obtenidas durante el desarrollo de este trabajo, destacando ventajas y desventajas de la arquitectura propuesta.

Capítulo 1

Conceptos de Personalización

Este capítulo está dedicado a introducir los conceptos y las características de las aplicaciones personalizadas, en especial se van a destacar las aplicaciones Web personalizadas.

La sección 1.1 explica el motivo de creación de la personalización, y ofrece una idea muy general de las características de la personalización. La sección 1.2 está dedicada a explicar qué es la personalización y los beneficios que ésta brinda. En la sección 1.3 se presentan dos características importantes de las aplicaciones personalizadas, y además los atributos de una aplicación que se toman en cuenta para personalizarla. La sección 1.4 se dedica a presentar particularidades de las aplicaciones Web personalizadas. Se describen qué aspectos de un sitio Web pueden ser personalizados, y se presentan dos maneras de clasificar a un sitio personalizado. Por último, en esta misma sección, se expone el tema de la privacidad de los datos de los usuarios, un tema muy delicado en este tipo de aplicaciones.

El objetivo de este capítulo es el de presentar conceptos claros sobre las aplicaciones personalizadas, para poder tener una buena base para comprender todo los temas tratado en los capítulos siguientes.

1.1 Motivación

El avance de las tecnologías de programación y la acentuada competencia existente en la Web, dio comienzo a una búsqueda de novedades para atraer a los usuarios de la misma. Este motivo provocó el origen de la creación de las aplicaciones Web personalizadas [1].

Anteriormente, los usuarios utilizaban Internet con fines de investigación, la misma brindaba gran asombro y satisfacción por la novedad y utilidad que ésta proveía, además de la invención que significa en esos días. Con el tiempo y la extraordinaria evolución que sufrió Internet, la calidad y la cantidad de usuarios que la utilizaba experimentaron un cambio profundo. Los usuarios dejaron de ser personas específicas con fines de investigación para pasar a ser personas comunes con diferentes intereses. Por dicha razón, las características e intereses de los usuarios sufrieron una gran transformación [2]. Estos nuevos usuarios se tornaron cada vez más exigentes e impacientes y su deseo consistía en obtener información útil de un sitio y que ésta sea proporcionada de manera inmediata sin que los mismos tengan que navegar mucho a través del sitio. Lo que sucede habitualmente, es que si el usuario no visualiza de una manera rápida información de su interés, se dirige a otro sitio y continúa saltando de un sitio a otro hasta hallarla. La creación de aplicaciones Web personalizadas ha logrado un cambio revolucionario con respecto a este suceso.

El objetivo de la personalización es adaptar el sitio a cada usuario de una manera personal, brindando a cada uno de ellos un ambiente más amigable, facilitando la búsqueda de información e interacción con el mismo. Con este tipo de aplicaciones se logra que el usuario encuentre información de su interés sin tener que navegar información irrelevante para él. Esto produce un interés total por parte del mismo, ya que en los casos que un usuario se encuentra en un sitio con demasiada información o se le dificulta encontrar cosas de su interés, abandona dicho sitio para ir hacia otro en el que se le brinde información y productos que llamen su atención. Además de la información, otra característica principal que brindan algunas de las aplicaciones personalizadas, es que el usuario puede armar el sitio conforme a sus preferencias. En estos sitios, el usuario puede seleccionar el aspecto visual del mismo así como también qué información desea ver. De esta manera, se comienza a adecuar el sitio al usuario particular logrando que el mismo se encuentre más cómodo al ver que el sitio se adapta a sus necesidades y es reconocido en el momento de ingresar [3].

Con la personalización se consiguen cubrir las necesidades de los usuarios de una manera más eficiente y más efectiva, logrando que las interacciones con el sitio sean más rápidas y más fáciles de realizar. Dichas características efectúan que se alcance el objetivo esperado, se incrementa la satisfacción del usuario, la probabilidad de que repita su visita y de que permanezca más tiempo en el sitio.

Para alcanzar la adaptabilidad que proporcionan las aplicaciones Web personalizadas, se debe almacenar y manipular información particular de cada uno de los usuarios que acceden al sitio. Esta información es obtenida en los momentos que el usuario navega por el sitio o cuando el usuario, en los casos que el sitio lo brinde, seleccione de manera manual información que es de su interés. Por dicho motivo, estas aplicaciones cuentan con contenedores de información de los usuarios, llamados perfiles de usuario, y además poseen lógica para manipular dicha información [4]. Esta lógica se encuentra comprendida por algoritmos de inferencia para realizar recomendaciones y adaptaciones al usuario, reglas de negocio para los usuarios, entre otros elementos que se serán explicados en las siguientes secciones.

Las aplicaciones deberán ser capaces de guardar toda la información posible de cada uno de los usuarios, se debe tener en cuenta que no toda la información es útil, y que tampoco se pueden guardar cantidades infinitas de información dado que debe ser almacenada en algún medio físico. Por lo cual, es importante evaluar y almacenar la

información que sea de utilidad filtrando los datos que no son útiles para el desarrollo de la personalización. La información obtenida de los usuarios es uno de los instrumentos más importantes a utilizar en las aplicaciones Web personalizadas, dado que, una buena personalización de la aplicación se logra con un profundo conocimiento de los intereses y preferencias del usuario. Cuanto más conocimiento se tenga sobre el usuario mejor será la adaptabilidad al mismo y mayor será su satisfacción.

1.2 ¿Qué es la Personalización?

Las computadoras son vistas por el común de la gente como máquinas no pensantes que no pueden formular más que las órdenes que se le envían a través del teclado y realizan procesamientos internos para resolver cálculos y otros procedimientos. Éstas solo son un medio para procesar información, utilizadas para el trabajo y entretenimiento, sin poseer ningún tipo de inteligencia propia. Las personas no están acostumbradas a que las computadoras les presenten sorpresas o cambios extraños, solo realizan lo que el usuario les indica que hagan. Con el concepto de personalización estas ideas han ido cambiando. Se le ha podido dar una cierta inteligencia a las aplicaciones, logrando captar los gustos y necesidades de los usuarios y ofrecerles de esta manera productos y servicios de su interés. Las aplicaciones personalizadas simulan un tipo de inteligencia, que reconoce a cada usuario ofreciéndole servicios y productos adecuados a las preferencias y necesidades de cada uno de ellos.

La personalización brinda al desarrollador la posibilidad de proveerle a cada uno de los usuarios diferente información sobre una misma aplicación. Es una herramienta que proporciona que el contenido, la interfaz y la funcionalidad de la aplicación se adapten a cada usuario particular. Esto se consigue adecuando estos aspectos mencionados en base a información sobre preferencias del usuario obtenidas en recorridos previos por la aplicación que componen un historial con la información del usuario conseguida al utilizar la aplicación en sesiones anteriores, o a través de la interacción en tiempo real. Las páginas que se crean para cada uno de los usuarios son creadas de manera dinámica interactuando con la información de los usuarios para poder conocer sus gustos y preferencias y adecuar cada una de las páginas a los mismos. Por este motivo, se deben almacenar características de la información que es buscada por el usuario en sesiones anteriores y se almacenan también los datos obtenidos sobre sus gustos y preferencias. Con la personalización se logra que las aplicaciones que se adaptan al usuario sean satisfactorias para el mismo, ya que el usuario ve la interfaz con colores seleccionados por él, el contenido se adecua a mostrarle solo información que puede ser de su interés, y además, existe la posibilidad de reestructurar el sitio como el usuario quiera, realizando las selecciones correspondientes.

Un ejemplo real que puede ser comparado con las aplicaciones personalizadas, es la relación que poseen los clientes con una tienda que se ubica cerca de sus casas, por ejemplo un video club. En estos lugares el cliente es una persona ya conocida en el lugar, es saludado y reconocido por el empleado del video amigo, es tratado como un cliente particular. En estos casos el empleado ya conoce sus gustos y puede recomendarle que llevar en el caso de no saber que alquilar. Conoce sus preferencias y sus gustos, basándose en las películas alquiladas anteriormente, además, los actores que pueden interesarle o sugerirle otras películas que pueden resultarle interesantes solo con conocer el general de sus gustos. Las aplicaciones personalizadas pueden ser comparadas con este caso real, en dónde cada usuario es reconocido y tratado como un cliente particular, y no como uno más que ha ingresado a la aplicación. Esto produce que el cliente se sienta cómodo y encuentre mucho más rápido la información buscada, y además como se marcó anteriormente en los sitios se le brindan recomendaciones basándose en los intereses que mostró el usuario en sus búsquedas y recorridos por el sitio. Lo que se

logra con las aplicaciones personalizadas es que el usuario se sienta atendido como un cliente particular.

Estas aplicaciones producen grandes beneficios tanto a los usuarios como a los proveedores de dichas aplicaciones. El usuario se siente más cómodo en un ambiente en donde es reconocido, donde el mismo contiene sus preferencias en cuanto a colores y estructura y puede encontrar la información con más rapidez, dado que solo ve información que puede ser interesante para él sin tener que navegar por un sitio lleno de información que no le interesa. Se logran cubrir las necesidades de los usuarios de manera eficiente y efectiva, haciendo las interacciones más rápidas y fáciles. Además el mismo se adapta a él, pudiendo brindarle información que quizás él no conocía que estaba pero la aplicación al conocer sus gustos se la muestra porque sabe que el usuario puede estar interesado, recomendaciones. Para los creadores de la aplicación, el realizar aplicaciones personalizadas tiene un costo mucho mayor que realizar una aplicación común, ya que las personalizadas deben proveer la funcionalidad de la aplicación en sí misma y la funcionalidad de la personalización. A pesar de esto, se obtiene un beneficio muy grande por el motivo que el usuario al ver información de su interés, queda mucho más complacido con la aplicación permaneciendo más tiempo en ella y queriendo retornar en otra ocasión. Si la aplicación está orientada a vender productos o servicios el beneficio es alto, ya que el usuario al ver lo que necesita adquiere el servicio o producto. Se puede realizar además, lo que en personalización se llama recomendaciones, se explicará más detalladamente en próximas secciones, que son productos o servicios recomendados al usuario basándose en sus intereses y preferencias, y que seguramente serán acertados y el usuario pueda interesarse y adquirirlos, logrando el creador más ganancia.

Esta inteligencia de las aplicaciones de la que se habla al principio de esta sección, se logra con la programación de la aplicación sumándole el mecanismo que se deben agregar para poder abarcar los aspectos que se encargan de la personalización. Por este motivo, las aplicaciones personalizadas son mucho más complejas que las comunes, pero el beneficio logrado es mucho mayor. No en todos los casos el uso de personalización es útil, se deberá evaluar dependiendo el sentido y objetivo de la aplicación que brinda si es conveniente emplear personalización o no.

1.3 Características de la Personalización

El objetivo de la personalización es el de asegurar que las personas reciban la información correcta en el tiempo justo. Para lograr este propósito, la tecnología de la personalización involucra software que aprende patrones, hábitos y preferencias de los usuarios basándose en que las personas pueden tener diferentes preferencias y prioridades cuando se trata de la información que necesitan.

Se deben destacar dos características importantes de la personalización. El primero, es que la personalización no se limita solo a aplicaciones Web, es aplicada en varios otros servicios con diferentes interfaces y por lo tanto con diferentes características, tanto en la utilización como en el funcionamiento de la misma. Como por ejemplo también es utilizada en los servicios de telefonía de voz usando reconocimiento de voz automático. El objetivo de la personalización en los servicios que la involucran es siempre el mismo, el de conocer y satisfacer los deseos y necesidades de cada uno de los usuarios. La segunda característica a destacar, es el hecho de que la personalización puede basarse en identificar cada usuario en particular, así como también, estar orientada a grupos de usuarios agrupados por algún factor en común que los distingue. Por ejemplo, la personalización del servicio de la aplicación puede estar diseñada y orientada a usuarios de cada provincia particular, lo cual ocasionaría que el servicio se encuentre personalizado de igual manera para las personas que pertenecen a la misma provincia. Estas dos características brindan una visión amplia de las posibilidades de aplicar la personalización a diferentes servicios, así como también aplicada de diferentes maneras.

Para acotar estas diversas variantes que existen y que han sido expuestas para tener un conocimiento de la amplitud de posibilidades existentes de aplicaciones personalizadas, se decidió acotar este trabajo a las aplicaciones Web, sin tener en cuenta el resto de las aplicaciones existentes, de esta manera se abarca solo un aspecto de la personalización, sin involucrar características de los demás servicios. Además, se hablará de la información de los usuarios y perfiles de usuarios, refiriéndose con esto a cada usuario como persona, pero sin dejar de marcar la posibilidad de que cada usuario pueda ser un grupo de ellas.

La adaptabilidad que se logra con la personalización abarca variados atributos de las aplicaciones a tener en cuenta para lograr personalizar aspectos que constituyen la aplicación final, las pantallas o páginas que ven los usuarios. Aspectos se refiere a contenido, estructura, interfaz, recomendaciones y búsquedas entre otros. Las aplicaciones pueden personalizar solo un elemento, combinaciones de ellos o todos los elementos posibles dependiendo del objetivo de las mismas. Algunos de los atributos que se emplean para realizar la personalización y la adaptabilidad al usuario son, las propiedades del usuario, el lugar de acceso, el dispositivo utilizado y el tipo de conexión.

Con respecto al usuario, se utilizan sus preferencias en cuanto al contenido, interfaz, diseño de la aplicación y también la información que se posee del usuario que fue obtenida en el momento de la registración. Se personaliza el contenido, que comprende la información que se le presenta al usuario, lo mismo sucede con la interfaz y el diseño de la estructura. Teniendo en cuenta la información del usuario se pueden personalizar muchos aspectos. Por ejemplo, en el caso que un usuario consulte por alguna película que desea ver en el cine. Al realizar la búsqueda, como ya se tienen datos sobre dicho usuario, como su dirección y código postal, la respuesta a dicha búsqueda le proveerá la película en cines cercanos a la zona donde habita dicho usuario. De esta manera, se brinda al mismo, no solo los cines y horarios en los que puede ver la película deseada, sino que se le brinda la adaptabilidad de solo mostrarle los cines cercanos a su domicilio; brindándole la posibilidad de consultar por otros cines. Además, se puede tener en cuenta por ejemplo su edad y demás datos que pueden ser útiles del mismo modo que la dirección para personalizar varios aspectos de la aplicación.

Con la información obtenida tanto de los registros de la computadora, como en los casos de las aplicaciones Web de los navegadores, se puede saber de qué lugar físico se encuentra logueada la persona, lugar de acceso. En estos casos la aplicación puede adaptar la información presentada al usuario. Por ejemplo, en una aplicación de noticias, se mostrarán las noticias pertenecientes al lugar de dónde se accedió a la aplicación, para informar al usuario lo que sucede a su alrededor, proporcionando la posibilidad de cambiar de lugar.

Para adaptarse a los diferentes dispositivos, la aplicación deberá de alguna manera darse cuenta si el usuario se conecta por ejemplo, de una computadora o de un teléfono celular, en tales casos los datos de la aplicación deben viajar en formatos diferentes. Los datos son adaptados tanto para un dispositivo como para otro sin que el usuario tenga conocimiento de la adaptación que se realiza. En este caso se personalizan el formato de los datos que terminan en el dispositivo.

Con respecto al tipo de conexión, se puede personalizar la información a mostrar dependiendo si la conexión es buena o mala. Por ejemplo, en el caso de que la persona se conecte a través de un modem y la conexión sea lenta, se puede disminuir la resolución de los gráficos de las pantallas a mostrar. En otro caso se mantendrá un nivel alto o medio de resolución.

Otras de las formas de personalización existentes, es el de personalizar búsquedas de información basándose en preferencias seleccionadas por usuario. Un ejemplo de este caso puede verse en http://labs.google.com/personalized/, en donde el usuario manualmente selecciona sus preferencias que quedan almacenadas en su perfil y las

búsquedas posteriores se basan en estos datos cuando se realizan. En la Figura 1.1 se muestran las categorías de temas a seleccionar, y sobre las mismas las opciones que se

pueden seleccionar para formar el perfil del usuario. Estos perfiles pueden ser cambiados por los usuarios en cualquier momento.

Se describieron aquí los atributos que colaboran para personalizar la aplicación. Existen muchos otros casos en los cuales se afectan los aspectos de las aplicaciones, acá solo se presentaron algunos ejemplos dado que las combinaciones que se pueden realizar son muy numerosas. En las secciones 1.4.1 y en el capítulo 4 la sección 4.4 se presentarán con más detalle aspectos de la aplicación que pueden ser personalizados.

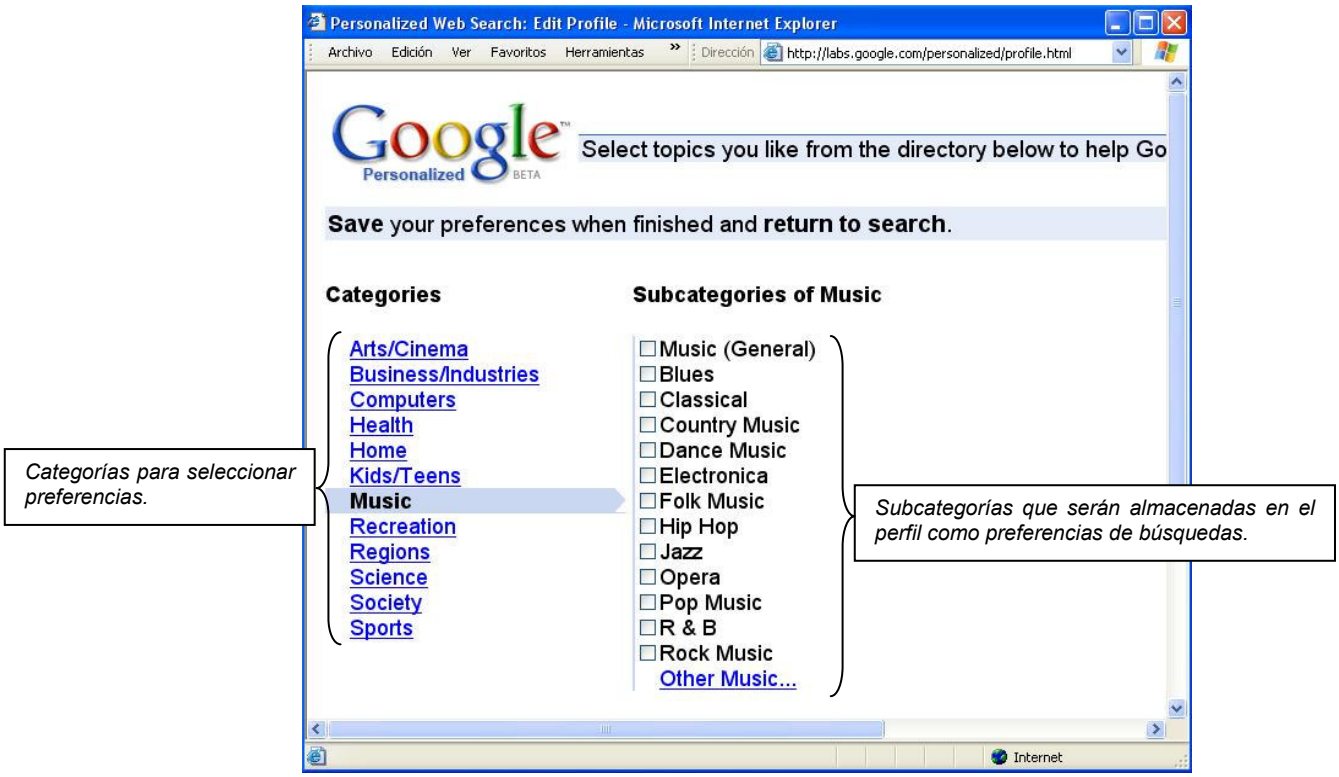

Figura 1.1: Selección de preferencias para el perfil de búsqueda.

Finalmente, se presenta una de las posibilidades para mejorar las aplicaciones personalizadas. Antes de crear la aplicación, se debe realizar un estudio del diseño a nivel visual y funcional que se va a brindar a los usuarios basándose en la funcionalidad de la aplicación y en los aspectos de la personalización que van a ser utilizados. No en todos los casos se puede obtener un buen diseño, ya que la personalización depende mucho de la interacción de los usuarios con la aplicación. Una de las posibilidades existentes que se proponen para mejorar las adaptaciones a los usuarios de las aplicaciones personalizadas, es la utilización de asistentes que colaboran con el diseñador para mejorar la organización y la presentación del sitio [5]. En estos casos, los diseñadores de los sitios añaden un asistente, que no es más que cierta programación que se agrega a la aplicación, el cual su función es la de procesar grandes cantidades de información que son utilizadas en el sitio, referentes a las interacciones de los usuarios con la aplicación. Este asistente tiene como objetivo sugerir ajustes útiles al diseñador basándose en la información obtenida. Procesan la información que obtienen con las visitas de cada uno de los usuarios, y al procesarla pueden capturar patrones comunes de navegación o de organización que utilizan. El uso de estos asistentes se adopta como ayuda para lograr una mejor adaptación a los usuarios; se logra mejorar la organización y

la presentación aprendiendo de los accesos que tienen al sitio. De esta manera, el diseñador registra los cambios que se deberían realizar para que se mejore o reacomode el sitio, logrando de este modo que los usuarios se sientan más cómodos dentro del mismo y con mayor facilidad de manejo.

1.4 Personalización en la Web

Los usuarios al ingresar en los sitios Web de comercio electrónico no personalizados, se encuentran con una diversidad de productos y servicios ofrecidos que posiblemente no sean de su interés o no todos sean de su interés, y algunos no les interese en absoluto. En estos sitios el usuario deberá ir recorriéndolo y seleccionando características dentro del mismo para encontrar la información buscada por el mismo. Los usuarios no reciben ningún tipo de ayuda del sitio para localizar la información que necesitan, deben basarse en el que sitio esté bien diseñado para que sea fácil y rápido encontrar la información buscada. Lo que generalmente contienen estos sitios es un buscador en donde se colocan palabras claves para que sean buscadas dentro de los mismos, y no siempre se logran los resultados deseados. Los usuarios se sienten perdidos sin ninguna ayuda ni solución para encontrar lo que necesitan [6], y seguramente el sitio se los provee, pero no encuentran la manera de hallarlo, logrando que el usuario se canse y desilusionado cambie de sitio.

Fundamentalmente, lo que se realiza en las aplicaciones Web personalizadas es tener diferentes adaptaciones de una misma aplicación. La información de las páginas varía dependiendo de las preferencias e intereses de los diferentes usuarios. La aplicación es la misma, solo que se adapta a cada uno de los usuarios mostrándoles información de su interés, como se muestra en la Figura 1.2. Estas páginas son creadas de manera dinámica con cada pedido realizado por el usuario, adaptando las páginas en el momento de la petición. Un ejemplo de este caso, se puede ver en Amazon, si se loguea un usuario cuya preferencia son los libros de suspenso, le mostrará al mismo y le recomendará libros de suspenso de un autor que haya comprado o consultado anteriormente, además le sugerirá libros de algún otro autor del mismo género. Cuando ingrese al sitio otro usuario, le mostrará libros de género de terror, en caso de que el interés del mismo sea ese.

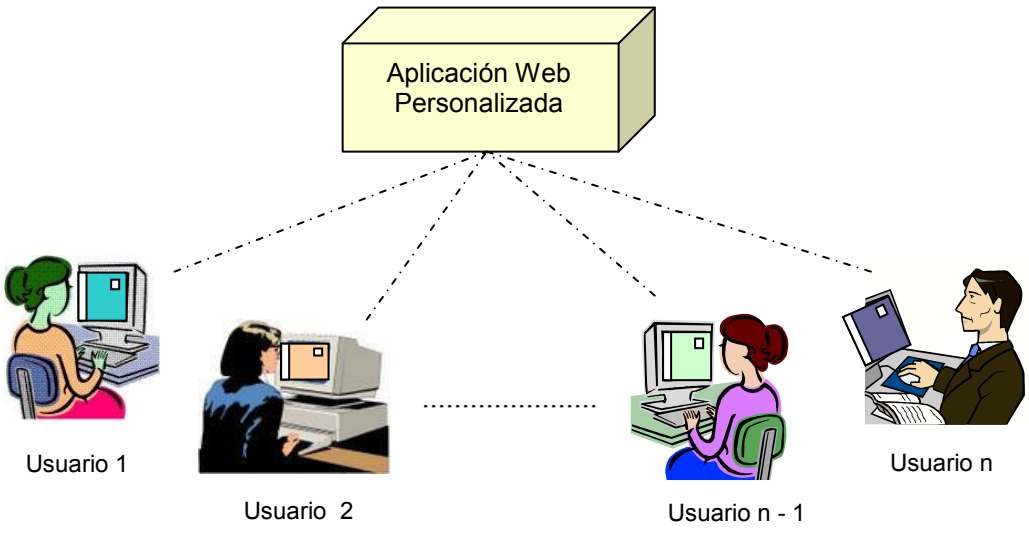

Figura 1.2: Diferentes páginas dependiendo de cada usuario.

Para ingresar en un sitio personalizado el usuario debe ser reconocido por el mismo. Para esto debe loguearse de alguna manera, o en el caso que sea la primera vez que se visita el sitio, se debe registrar ingresando sus datos. En algunas aplicaciones Web personalizadas se personalizan algunos aspectos de la misma sin que el usuario se haya registrado previamente. Esta personalización se basa en la información que puede ser obtenida de los navegadores, sin tener ninguna interacción con el usuario. Se puede saber por ejemplo la preferencia del idioma de la persona, sin haber interactuado nunca con la misma. Así la primera vez que el usuario entra en el sitio ya se presenta una adaptación al mismo. La registración tiene como propósito que el usuario sea reconocido dentro de la aplicación para almacenar sus datos y utilizarlos para adaptar el sitio a sus preferencias y necesidades. La primera acción que debe realizar la aplicación es el reconocimiento de la persona para realizar la personalización correspondiente, este tema de reconocimiento de los usuarios será ampliado en la sección 2.3. En general, la primera vez que el usuario se loguea al sitio no lo visualiza personalizado, visualiza el sitio por defecto. Luego de recorrerlo y navegar por el mismo, las preferencias del usuario son guardadas para que en su siguiente ingreso al sitio, la aplicación utilizando estos datos almacenados logre un ambiente más amigable, el sitio reconoce al usuario y se adapta al mismo. Cuanta mayor cantidad de información se adquiera del usuario, por la inferencia que se realiza al navegar o por el ingreso específico del usuario, se obtiene un conocimiento mayor y la adaptación del sitio será más precisa con respecto a sus preferencias e intereses. En la sección 2.4.3 se desarrollará el tema de obtención de la información de los usuarios.

Se debe destacar por último, una particularidad que presentan los sitios Web personalizados. Es muy frecuente que estas aplicaciones presenten la sección que constituye la personalización en una sección separada del resto del sitio “my.com”, como se muestra en la Figura 1.3. Como se destaca en [7], esto provoca en los usuarios la sensación de que la personalización no se encuentre integrada al sitio logrando una gran desventaja ya que la personalización es vista como una parte separada del sitio. Esto provoca además que gente que no tenga conocimiento de la existencia de la personalización no utilice esta sección tan significativa. Existen otros sitios que proveen la personalización incluyéndola dentro del sitio, pero son muy pocos los que han adoptado esta costumbre. Por lo cual está visto My como una representación de la sección personalizada.

Figura 1.3: Uso del My en los sitios personalizados, My eBay.com.

1.4.1. Características que pueden Personalizarse

La personalización realizada en las aplicaciones Web abarca una gran cantidad de elementos que pueden ser adaptados a los usuarios. Basándose en los perfiles de usuario y las reglas de negocio, las páginas se van adaptando a los usuarios dinámicamente. Cada aplicación personaliza los elementos que crea necesarios en su aplicación, no todas las aplicaciones personalizan lo mismo, dependerá de qué beneficios quiera brindarle a sus clientes y sea útil para su provecho y costos [8]. Las páginas pueden ser pobremente personalizadas o tener varios de los elementos personalizados, lo cual indica que el usuario se sentirá más a gusto en estos sitios porque contienen más adaptaciones a sus preferencias.

En relación al aspecto visual del sitio se pueden personalizar varios elementos, se pueden seleccionar los colores de fondo de las pantallas, de los títulos, subtítulos, enlaces, de los textos, los tipos de letras y los tamaños. En algunos sitios se proveen temas de colores que los usuarios pueden seleccionar para verlos en las páginas. Un ejemplo de estos casos se puede encontrar en http://my.yahoo.com/. En general, este tipo de información es seleccionada manualmente por el usuario, se marcan y seleccionan los colores de los elementos y estos quedando guardados para una próxima visita.

La personalización de contenido, en cuanto a información, es un poco más complicada de realizar ya que se deben tener en cuenta varios atributos internos del manejo de la aplicación para personalizar la información que se muestra a los usuarios. Se personalizan los productos y servicios dependiendo de los intereses del usuario, se puede personalizar además el contenido mismo de cada información, así como agregar ofertas y aplicar descuentos dependiendo del tipo de cliente. En estos casos, ciertas aplicaciones personalizan el contenido de los precios haciendo descuentos a los clientes asiduos, y quizás también ofertándoles productos a bajo precio. Otro ejemplo que puede citarse es, en caso de una tienda que vende productos al por mayor y por menor, diferente será la información en cuanto al precio que exhibirá si ingresa al sitio un cliente mayorista o un cliente minorista. Muchas aplicaciones realizan descuentos en las

festividades, y otras posibilidades más. Se puede ver también que en ciertas tiendas electrónicas personalizan las recomendaciones según las preferencias. Otra característica de la personalización de contenido es la traducción de las páginas al idioma del usuario, se conoce de alguna manera el idioma del usuario y se adaptan las páginas al mismo, logrando de este modo que el usuario se sienta cómodo con el idioma utilizado en la aplicación. El contenido también es personalizado como se explicó en la sección 1.3 en caso de que la conexión de Internet sea mala, adaptando los gráficos a una resolución menor. Existen un montón de variantes que pueden provocar que el contenido varíe, pero es más frecuente que éste cambie por un procesamiento interno que la aplicación realiza para adaptar el contenido y no tanto por una selección del usuario. Sí, se da el caso de que los usuarios seleccionarán el contenido eligiendo categorías de preferencias sobre gustos como sucede en http://labs.google.com/personalized/.

Otra característica que puede ser personalizada es el diseño o estructura de las páginas, la adaptación de los elementos que contienen información dentro del sitio. Se puede hablar en estos casos de bloques de información que componen las páginas. La información está distribuida en bloques de diferentes temas, por ejemplo un bloque puede contener información del pronóstico del tiempo. Este aspecto a personalizar no es muy frecuente de encontrarlo en los sitios personalizados, pero es utilizado y resulta muy útil en los sitios que proveen bloques de información que se pueden decir que son estáticos o definidos de antemano, como puede suceder en un sitio de noticias, provee bloques con temas como el pronóstico, los titulares, internacionales y espectáculo, entre otros. Lo que se realiza con esta característica es que el usuario seleccione qué información desea ver y cual no, pero no como fue explicado anteriormente refiriéndose a nivel de contenido sino que se seleccionan explícitamente qué bloques de información se desea que estén visibles en las páginas y cuales no, según lo desee el usuario. Asimismo, se puede personalizar el contenido de cada uno de los bloques, por ejemplo en el caso del pronóstico se pueden elegir de qué ciudades o países le interesa al usuario tener información. De esta manera, el usuario estructura la página a su gusto, y solo va a ver lo que es de su interés sin tener a la vista bloques de información que nunca va a navegar. Este aspecto es muy útil y aprovechado en sitios con mucha información logrando filtrar de este modo información relevante. Además de poder seleccionar los bloques a visualizar, otra adaptación que se provee es la reubicación de los bloques de información, el usuario puede elegir la ubicación de los bloques dentro de la página, logrando de este modo que se pueda manejar casi todo lo referente al diseño del sitio. Un ejemplo de esta característica de personalización puede ser vista en http://my.yahoo.com/. Este aspecto es un patrón de personalización que será expuesto en el capítulo 4, sección 4.4.3.

La última característica a destacar que puede ser personalizada es la navegación. Los links que se encuentran en el sitio y que se muestran al usuario se dirigirán a diferentes páginas dependiendo de las preferencias de los mismos. Lo que se logra al alterar la dirección de los links es cambiar las páginas que navega cada usuario, y como consecuencia la topología de navegación puede ser diferente para los usuarios. Solo se expone una idea general de esta característica ya que será presentada en detalle en el capítulo 4 como un patrón de diseño de las aplicaciones personalizadas.

Como se puede observar existen una gran variedad de características que pueden ser personalizadas, y a su vez, estas mismas depender de un gran conjunto de variables. Para cada característica se tienen variables diferentes que pueden depender del usuario o de las reglas de negocio de la aplicación, así como también de factores externos como puede ser el tipo de conexión, entre otros. Por estas variedades de posibilidades, la personalización es una herramienta muy utilizada pero que posee sus complicaciones, requiere inversión de tiempo en diseño y selección de las características a personalizar y un trabajo exhaustivo para poder poner en producción la aplicación de una manera correcta y produciendo los beneficios correspondientes.

1.4.2. Customización y Personalización

Customización y Personalización son dos términos que se encuentran muy relacionados con la obtención de información de los usuarios en las aplicaciones personalizadas. Son muy utilizados refiriéndose a estas aplicaciones, tienen diferentes significados y cabe destacar esta diferencia.

Customización ocurre cuando el usuario tiene control sobre la elección de sus preferencias. El usuario puede configurar una interfaz y crear un perfil manualmente, agregando y eliminando elementos del perfil. En este caso, el usuario explícitamente selecciona las opciones de su interés. El control del aspecto visual y del contenido es explícitamente manejado por el usuario, el mismo se encuentra involucrado activamente en el proceso de recolección de sus datos y tiene el control sobre ellos. Un ejemplo de customización puede verse en http://labs.google.com/personalized/.

En cambio, en la personalización, el usuario está involucrado de una manera pasiva o con un grado mucho menor de control sobre la elección de sus preferencias. Es el sistema el que monitorea, analiza y recolecta el comportamiento del usuario, sin que el usuario tenga conocimiento de que se está recolectando información sobre él. Por ejemplo, se puede guardar información sobre el recorrido de las páginas y decisiones que realizó el usuario en su permanencia en la aplicación. En este caso, el usuario no tiene control sobre la información que se recoge sobre él, contrapuesto a lo que sucede en la Customización. Estas dos definiciones de Customización y Personalización se basan en las definiciones presentadas en [9].

Basándose en estas dos definiciones, se puede realizar una clasificación con respecto a los sitios Web personalizados. Pueden ser clasificados como adaptables (relacionado con el término customizable) o adaptativos (relacionado con el término personalizable). Si el sitio es adaptable, el usuario tiene control sobre sus preferencias y participa activamente de la elección de sus intereses y puede manejar el perfil, agregando y eliminando aspectos del mismo que colaboran con la personalización. En cambio, existen otros sitios que se denominan adaptativos en los que el sistema monitorea y es el encargado de seleccionar y almacenar la información que necesite del usuario para agregar al perfil del mismo y así lograr la personalización del sitio. También existen sitios que combinan las dos posibilidades, logrando una mayor potencia de recolección de información del usuario que será almacenada para que luego sea utilizada para personalizarlo cuando sea visitado la próxima vez.

Como se acaba de presentar, estos dos términos pueden ser utilizados por las aplicaciones de manera individual o ser combinados en la recuperación de información de los usuarios en las aplicaciones personalizadas, como se explicará con más detalle en la sección 2.4.3.

1.4.3. Personalización y Privacidad

En las aplicaciones de hypermedia personalizadas, especialmente en las aplicaciones Web personalizadas, se recolectan y procesan datos de los usuarios. Por causa de estas acciones, surge un tema muy delicado que preocupa a la gran mayoría de los usuarios que es la privacidad que existe sobre la información que se obtiene y almacena sobre ellos. El tema de la privacidad de los datos también se da en las aplicaciones comunes, pero hay que tener en cuenta que en las aplicaciones personalizadas se recoge muchísima más cantidad de datos sobre los usuarios. Se han realizado estudios sobre las preocupaciones que se producen en los usuarios al navegar en las aplicaciones y se concluyó que los usuarios se preocupan, por las amenazas a la privacidad de sus datos cuando utilizan Internet, sobre los datos que se recogen mientras éstos navegan por los sitios, como sucede en las aplicaciones adaptativas, en donde el sistema monitorea obteniendo datos del usuario, y se encuentran más intranquilos a la hora de divulgar información personal. Los resultados con los porcentajes precisos pueden ser vistos en

[10]. Como consecuencia de estas preocupaciones, una situación crítica para los sitios que se produce frecuentemente, es que cuando los usuarios deben ingresar sus datos personales, éstos abandonan el sitio. Una posibilidad para lograr un poco de privacidad y conformidad de los usuarios, es que los sitios pidan permiso para usar su información personal, y que además, a cambio de la misma, se les brinde cosas valuables utilizando sus datos, como pueden ser las ventajas que provee la personalización en las cuales el usuario brinda su información recibiendo beneficios a cambio.

Para evitar estos conflictos que comenzaron a surgir en las aplicaciones respecto de la privacidad de los datos, se crearon leyes que protegen la información de los usuarios [11]. Protegen la información que los identifica, así como también la información que es almacenada que no identifica al usuario, pero que realizando un poco de esfuerzo basándose en los datos recolectados el usuario puede ser identificado. Cuando los usuarios son registrados en un sitio se le presentan las reglas y normativas de privacidad que posee, está en el usuario aceptarlas o no, pero además se sabe que sus datos son amparados por la ley [12]. Esto evita que los datos del usuario sean publicados, vendidos a otras compañías o utilizados de alguna mala manera. Las leyes aplicadas a las aplicaciones de hypermedia personalizadas, no solo afectan a las condiciones sobre las cuales la información personal es recolectada, sino también, se aplican a los métodos que son utilizados para procesar dicha información. Existen restricciones que, para proteger la información de los usuarios, afecta a la funcionalidad interna de las aplicaciones de hypermedia personalizadas. Un ejemplo de lo mencionado sucede en German Teleservices Data Protection Act y en European Data Protection Directive, como se encuentra citado en [10] y se enumeran dichas restricciones.

Existe una organización que tiene como propósito la construcción de estándares para la utilización de la personalización, llamada Personalization Consortium. Su objetivo es no destruir la personalización, pero sí asegurar las características de privacidad para que los usuarios utilicen las aplicaciones con seguridad de que sus datos se mantengan en confidencialidad. Por ejemplo, regulan que los procesos de las tiendas de compras online sean seguros y más convenientes para los usuarios. Esta organización internacional está formada por miembros que son inscriptos voluntariamente que pueden ser compañías u otros, que participan en foros y discusiones sobre personalización. Se puede obtener más información de esta organización en http://www.personalization.org/.

Para cerrar la sección referida a la privacidad, a continuación se definen algunos de los puntos más importantes a tener en cuenta con respecto a la privacidad de los datos:

- Proveer a los usuarios información sobre todos los datos que se recolectan sobre ellos, los procesos y transferencias de los mismos, e indicarles los propósitos de cada una de las acciones.
- Obtener informados a los usuarios y pedirles el consentimiento para procesar los datos.
- Permitir a los usuarios inspeccionar, bloquear, rectificar, modificar y borrar datos que fueron provistos por ellos, así como también los que fueron inferidos por el sistema.
- Proveer mecanismos de seguridad sobre los datos de los usuarios.

Si estos puntos se encuentran definidos en la aplicación, el usuario se sentirá más seguro con respecto a la privacidad de sus datos, y de este modo, dará su información personal sin dudar de sus fines y cuidados.

Capítulo 2

Características de las Aplicaciones Personalizadas

Este capítulo presenta aspectos y características propias de las aplicaciones personalizadas, presentando aspectos más específicos que el capítulo anterior y puntualizando temas delicados en los que se deben tomar buenas decisiones a la hora de elegir una solución, dado que si esto no sucede y se pasan por alto decisiones importantes, se podría ver afectada la performance de la aplicación.

Comienza con una introducción a las aplicaciones Web personalizadas, se presentan conocimientos generales de sus funcionalidades y características, que son desarrollados en la sección 2.1. En la sección siguiente 2.2, se describe el dinamismo de la información de los usuarios y la creación dinámica de las páginas, que desechan toda tecnología estática para ser utilizada en este tipo de aplicaciones. La sección 2.3 se ocupa de presentar cómo se efectúa la identificación de cada uno de los usuarios, qué modalidades se pueden utilizar, anexando ejemplos de dichas modalidades, además, se describen las ventajas y desventajas de cada una de ellas. La sección 2.4 trata sobre una de las secciones más importantes de las aplicaciones personalizadas, el perfil de usuario, describiéndose qué representa y qué funcionalidad cumple en estas aplicaciones. Esta sección se focaliza en exponer los aspectos de manejo y almacenamiento de la información de los usuarios que se debe mantener y manejar para poder realizar la personalización. Luego se tratan algunas cuestiones importantes con respecto a los perfiles, como son la de permitir a los usuarios la actualización de su información, dentro del perfil de usuario, y una cuestión muy delicada como es la seguridad y privacidad de los datos. Por último, la sección 2.5, explica y describe el tema de las reglas negocio, parte fundamental de las aplicaciones de comercio electrónico.

El propósito de este capítulo es el de exponer temas fundamentales para las aplicaciones personalizadas. Estos temas son presentados desde un punto de vista más técnico, poniendo foco en qué elementos se necesitan para desarrollar aplicaciones personalizadas y describiendo cuales son las funcionalidades de dichos elementos.

2.1 Introducción

Para el desarrollo de las aplicaciones personalizadas no se puede tener bajo ningún concepto un modelo para cada usuario, dado que es absurdo de implementar e imposible de mantener. Por este motivo, se tiene un modelo compuesto por las características de la aplicación y las características de los usuarios, dicho modelo realiza el manejo de la información propia de la aplicación, así como también, manejo e interacción de la información de los usuarios para lograr la adaptación a cada uno de ellos que es requerida por la personalización.

Las características y funcionalidades más importantes en la personalización son la identificación e individualización de cada uno de los usuarios dentro de la aplicación y la adaptación a cada uno de ellos. Para lograr estas adaptabilidades que introduce la personalización, se debe realizar fundamentalmente el almacenamiento y manejo de la información de los usuarios, consiguiendo que los mismos reciban resultados acertados, así como también, logrando que los tiempos de respuesta sean razonables. La personalización de la aplicación se obtiene con el dinamismo que introduce la adaptación a los usuarios, orientando el dinamismo a dos cuestiones. La primera, con respecto a la información de los usuarios que se va recolectando y actualizando para obtener sus preferencias e intereses, y la otra cuestión, es el dinamismo que introduce armar cada página en el momento que es solicitada sin tener un previo armado de la misma, dado que, las preferencias de los usuarios cambian de unos a otros y las páginas deben ser adecuadas a cada uno de ellos. Con respecto a la información que se debe conservar sobre cada usuario se tiene en las aplicaciones lo que se denomina perfil de usuario, que no es más que una unidad donde se almacena toda la información obtenida de los usuarios. El tema perfil de usuario se explicará con más detalle en la sección 2.4.

En las aplicaciones personalizadas no alcanza únicamente con recolectar y almacenar información de los usuarios, se deben realizar otros procesos para lograr traducir las preferencias de los usuarios en adaptaciones del sitio con respecto a la estructura, el contenido y la interfaz de las páginas. Estos procesos involucran la resolución de qué links incluir en la página, los contenidos que se presentarán, la estructura que van a contener y el aspecto visual que se proporcionará. Cada uno de estos procesos se encuentra sujeto a la complejidad que deben poseer para resolver su propósito, como por ejemplo, para resolver el contenido de las recomendaciones un proceso que se realiza y es muy importante, es la ejecución de algoritmos[1] que tienen por objetivo resolver los distintos productos o servicios que pueden interesarle al usuario [13]. Por ejemplo, existen algoritmos que se ejecutan, que se basan en los gustos que coinciden entre los usuarios, y de este modo, sugieren productos logrando resultados óptimos y adaptaciones que se supone le serán de interés al usuario ya que se basan en otro usuario con preferencias similares. Además, los sitios de comercio electrónico, así como cualquier negocio de venta de productos o servicios, tienen reglas de negocio establecidas. Éstas pueden ir variando a lo largo del tiempo de vida de la aplicación y se basan en políticas de descuentos y ofertas que benefician a todos los usuarios o a un grupo de ellos, dando como resultado la realización de otros procesos que se encargan de ejecutarlas para lograr otras adaptaciones. Se profundizará en el tema de reglas de negocio en la sección 2.5.

Como conclusión a lo expuesto en esta sección, se presenta un breve resumen de las actividades que implican las aplicaciones personalizadas. Para lograr la adaptación necesaria, cada aplicación se debe basar principalmente en el corazón propio de la

1 Estos algoritmos son muy complejos y suelen basarse en métodos matemáticos para ser resueltos, es un tema muy importante que es tratado en los artículos referentes a las aplicaciones personalizadas. Se realiza el estudio de algoritmos óptimos y de respuesta rápida.

aplicación, su funcionalidad y su objetivo, adicionándole información y realizando los procesos que sean necesarios para conseguir con éxito la personalización de la misma. Estos procesos se basan fundamentalmente en la recolección y mantenimiento de los datos de los usuarios, y en la funcionalidad misma de la aplicación efectuando interacciones con los perfiles de usuarios y aplicando las reglas de negocio que posee, logrando de esta manera la adaptación de las páginas finales a cada uno de los usuarios.

2.2 Dinamismo de la Información

En los principios cuando se crearon las aplicaciones Web, solo eran páginas que se mostraban a los usuarios, las cuales eran estáticamente creadas, se almacenaban en el servidor y eran enviadas al cliente cuando eran requeridas por los mismos. Con la evolución de las tecnologías, esta manera de desarrollar e implementar las aplicaciones Web fue cambiando, pudiendo mejorar la calidad del diseño de los sitios y logrando que los sitios sean cada día más dinámicos, provocando como consecuencia que la implementación sea mucho más compleja [14]. Estas nuevas tecnologías lograron que las partes que componen una aplicación se fueran desacoplando, obteniendo como resultado que las tareas de desarrollo y mantenimiento de cada una de las partes se efectúen de una manera más sencilla y fácil. Con la creación de las aplicaciones personalizadas, la idea de crear páginas estáticas queda descartada, dado que no se pueden tener las páginas previamente creadas para cada uno de los usuarios del sitio, sino que se deben crear y adaptar en tiempo de ejecución. Por dicha razón, no todas las tecnologías existentes en el mercado son capaces de soportar este tipo de aplicación, solo son utilizadas las que proveen la creación dinámica de las páginas. Esto, además, pudo ser logrado por el motivo de que las tecnologías fueron evolucionando permitiendo el completo dinamismo de creación de las páginas. Para proveer personalización se deben utilizar herramientas y metodologías por las cuales se pueda proveer dinamismo del sitio, las herramientas estáticas quedan totalmente descartadas en este tipo de aplicaciones.

La creación dinámica de las páginas trae como consecuencia que la performance de la aplicación se encuentre muy afectada, dado que, para lograr este dinamismo, se deben realizar ejecuciones propias de la aplicación que deben interactuar con el perfil del usuario para obtener sus preferencias y de esta manera lograr la personalización adecuada. Otro aspecto que compromete la performance de la aplicación, es la recolección de información de los usuarios mientras éstos navegan por el sitio, lo que causa tráfico en la red y procesamiento de almacenamiento de manera casi continua. Por todos estos motivos, se deben tener en cuenta diferentes aspectos de la aplicación que pueden ser críticas con respecto al tema de la performance y se deben tomar buenas decisiones para lograr que no sea afectada la calidad del sitio en cuanto a su rapidez de respuesta y procesamiento de información de los usuarios. Por ejemplo, se puede citar el caso de la aplicación MyYahoo, en donde se dispone de una aplicación importante, en cuanto a que posee millones de usuarios que se conectan y lo navegan, lo cual implica almacenar información de los mismos. En esta aplicación, se utiliza una base de datos especialmente diseñada, donde el tiempo de respuesta de las transacciones realizadas en la base de datos es extremadamente escaso, con el objetivo de mejorar la performance [15]. Además, le han agregado a la misma base de datos algunas otras características, como la replicación de datos, para lograr una mejor respuesta. Esto lo realizan basándose en que para la gente propietaria de la aplicación es de mucha importancia la velocidad, como así también, la eficiencia para complacer las expectativas de sus usuarios. Otra de las posibilidades que se encuentran disponibles para mejorar el tema relacionado con la performance, es la utilización de caching. Esto se utiliza para reducir el tiempo de latencia y cargado de las páginas. Esta técnica propone que las páginas o partes de ellas

que son frecuentemente accedidas deben ser almacenadas en el browser del usuario o en los servidores proxys y en los servidores caches [16].

Estos datos se exponen en esta sección para exhibir la existencia de otras dificultades que se presentan en las aplicaciones personalizadas que quizás no son tenidas en cuenta a la hora de pensar en realizar una aplicación de este tipo. Sin embargo, son de vital importancia para el funcionamiento eficiente de la aplicación ya que en los casos que el tiempo de respuesta de las páginas es muy alto los usuarios se cansan y abandonan el sitio.

2.3 Identificación de los Usuarios

La primera acción que se debe realizar en las aplicaciones personalizadas es identificar qué usuario está accediendo a la aplicación, y de este modo, lograr la personalización adecuada a él [17]. Esta identificación puede ser realizada una vez que el usuario se encuentre registrado en la aplicación, o sea, cuando por lo menos haya visitado una vez el sitio y haya realizado el proceso de registración, ya que si este proceso no fue efectuado no es conocido por la aplicación y no se tiene información del mismo. Si sucede esta situación, se le solicita que realice el proceso de registración para que la persona pase a ser un usuario reconocido por la aplicación.

Existen dos modalidades de identificación de los usuarios. Una de las modalidades, es que el usuario necesite ingresar un usuario y una contraseña para ser reconocido por el sitio, lo que se denomina loguearse en la aplicación o realizar un loguin [18]. La otra modalidad de identificación, es que el usuario sea reconocido por la aplicación de manera automática sin tener que realizar el loguin. Para ambas modalidades, en algún momento el usuario debe realizar el proceso de registración [19], el cual consiste en llenar una especie de formulario con sus datos personales. En la primera modalidad de identificación, se le pide además, que registre un nombre de usuario y una contraseña para que en sus próximas visitas al ingresar estos datos sea reconocido. Este caso es presentado en la Figura 2.1, donde se muestra la página de MyYahoo y se solicita el ingreso del usuario y la contraseña para entrar en la aplicación personalizada o clickear el link para registrarse.

Figura 2.1: Ingreso a MyYahoo con Usuario y Contraseña.

En la otra modalidad, es la aplicación la encargada de reconocer quien es el usuario que navega por la misma utilizando diferentes técnicas, como se describen a continuación. Una de las técnicas que son utilizadas para lograr la identificación de los usuarios es el empleo de las cookies [8]. Cookie no es un término nuevo en Internet sino que es utilizado desde años atrás, solo que en este caso se orienta al reconocimiento de usuarios. Es un pequeño paquete enviado desde la aplicación Web, que es almacenado localmente en la máquina del usuario y retorna información respondiendo a los pedidos del servidor. Este paquete contiene información que identifica al usuario y es accedido por el servidor mientras que el usuario interactúa con la aplicación sin que la persona tenga conocimiento de dicha acción, de este modo la reconoce sin tener que solicitarle que se realice el loguin. Se puede ver un ejemplo de este caso en el sitio Amazon, como se observa en la Figura 2.2, al ingresar al sitio el usuario automáticamente es reconocido sin realizar ningún tipo de loguin. La gran ventaja de utilizar este mecanismo para la identificación de los usuarios, es que los mismos son reconocidos por la aplicación sin que se deba pedirles ingresar un usuario y una contraseña. Es importante presentar también las desventajas de esta técnica. Para que este mecanismo funcione, el usuario deberá tener activada la opción que permite almacenar cookies en su máquina, en caso contrario, este procedimiento no puede ser llevado a cabo. Otra situación que puede ocurrir, es que se realice la eliminación de cookies en la máquina del usuario lo que producirá que se pierda la cookie que informaba a la aplicación su identificación, debiendo ingresar como un nuevo usuario la próxima vez que se desee acceder al sitio. Además, esta identificación solo es válida mientras que el usuario ingrese a la aplicación desde una misma máquina, ya que el almacenamiento de la cookie se realiza en la máquina del cliente, si se produce el caso de que el usuario acceda a la aplicación desde otra máquina, no será reconocido y no verá el sitio personalizado con sus características debiendo registrarse nuevamente. Como última desventaja, se presenta el tema de que diferentes usuarios utilicen una misma máquina, en este caso, si no existen diferentes sesiones para cada usuario, serán identificados por la aplicación siempre como un mismo usuario a pesar de no ser la misma persona la que realiza el acceso.

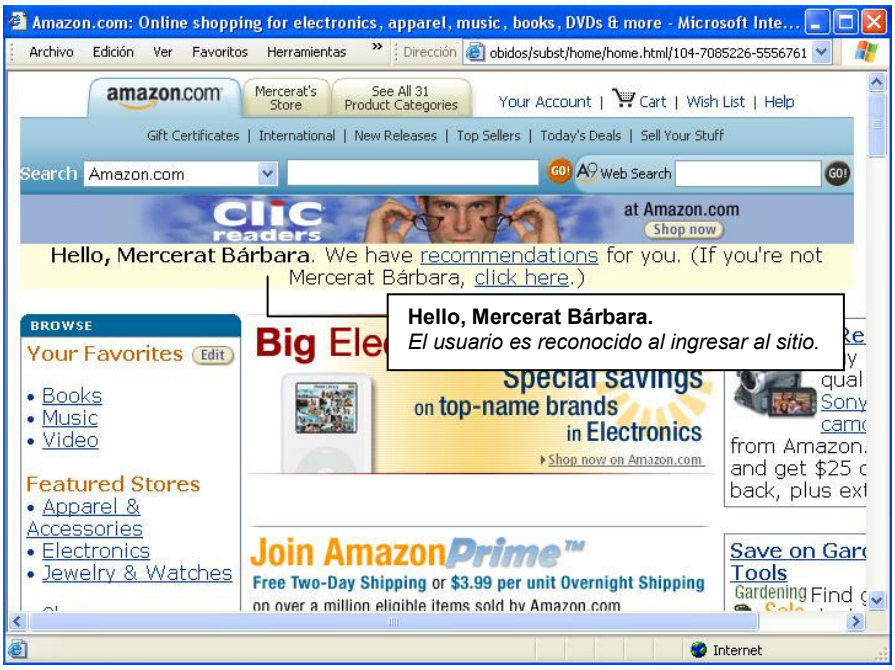

Figura 2.2: El usuario es reconocido automáticamente por la aplicación.

Otra técnica para identificar a los usuarios, con la misma modalidad de la presentada anteriormente, es a través de transacciones en la Web utilizando identd[2], es un protocolo de identificación que provee información para determinar la identificación del usuario a partir de la conexión particular de TCP (Transmission Control Protocol) [19]. Brinda un número de puerto de TCP y retorna un string de caracteres, el cual identifica el dueño de la conexión, o sea al usuario que está utilizando la aplicación. Con esta opción se asume que para la misma dirección IP (Internet Protocol) corresponde un mismo usuario. La ventaja de esta técnica es la misma que se presenta con la utilización de cookies y así como éstas, tiene desventajas, unas muy similares a las que se producen en dicha técnica. Si el usuario accede a la aplicación de una máquina diferente, no podrá ser identificado ya que la dirección IP de la nueva máquina no será la misma. Además, si la máquina es utilizada por diferentes usuarios no se puede identificar a más de uno de ellos, a pesar de que éstos utilicen sesiones de usuarios diferentes dado que la máquina posee para todos la misma dirección IP. Otros casos que surgen con esta técnica, es que una misma IP se le puede dar a un router el cual conecta más de una máquina a la vez logrando de esta manera con una dirección la conexión de varias máquinas utilizadas por diferentes usuarios de diferentes máquinas, y de este modo, la identificación de los diferentes usuarios es imposible de realizar. Otro tipo de conexión que existe es el de conectarse a través de un dial up, que lo que realiza es otorgar IPs dinámicamente, por lo cual las IP de cada máquina van variando y si la identificación se realizaba por número de IP pudo haber sido usada por un usuario y luego por otro. Por todos estos inconvenientes, se puede concluir que es más factible identificar a usuario con cookies que con la técnica de identd.

La gran diferencia que se puede destacar entre estas dos modalidades de identificación, es que la aplicación que utiliza loguin puede ser personalizada independientemente de la máquina de la cual acceda el usuario, con el nombre de usuario y la contraseña la persona es identificada unívocamente. Cuenta, sin embargo, con la desventaja de que el usuario para ser reconocido debe ingresar su usuario y contraseña,

2 Este protocolo de identificación se encuentra especificado en RFC 1413.

lo cual no siempre es tan sencillo, dado que existen usuarios muy temerosos de que sus datos puedan ser manipulados por gente externa a la aplicación, temiendo que la máquina que usen no sea segura para su conexión. Otra cuestión es que el usuario debe recordar su nombre de usuario y contraseña, para que esto siempre suceda o para prevenir el caso en que los usuarios se olviden usuario o contraseña, se debe proveer de un mecanismo para lograr comunicarles estos datos.

En cada aplicación se deberá decidir qué tipo de registración se realizará y con cual técnica, pero esta decisión dependerá básicamente de las características que presente la aplicación, evaluando las ventajas y desventajas de cada una de las modalidades de identificación y técnicas a utilizar. Existen aplicaciones que presentan las dos modalidades de identificación que fueron presentadas. El objetivo de las mismas es poder reconocer a los usuarios de una manera u otra, en el caso por ejemplo, que si el usuario tiene restringido el acceso de cookies a su máquina podrá ingresar a la aplicación utilizando su nombre de usuario y contraseña.

2.4 Perfil de Usuario

Como se fue explicando a lo largo del capítulo 1, las aplicaciones personalizadas deben mantener información sobre preferencias e intereses de sus usuarios para lograr realizar la personalización de manera adecuada, distinguiendo las diferentes preferencias e intereses de los usuarios o grupo ellos. Por este motivo, surge la creación de una pieza fundamental de las aplicaciones personalizadas que se denomina Perfil de Usuario [20]. Este puede ser visto como una estructura de datos que almacena información de cada uno de los usuarios, la aplicación contiene una de estas estructuras por cada usuario registrado. Esta estructura es creada para poder almacenar, mantener y manipular información de cada usuario. Debe poseer las características de ser flexible y de fácil extensión por futuras modificaciones, ya que los aspectos a personalizar pueden ir variando y quizás se deba almacenar más información que la que se almacenaba o dejar de almacenarla para lo cual quizás se desee eliminar cierta parte del perfil. La estructura que se le proporcione al perfil del usuario debe ser pensada con respecto a dos aspectos, qué información se deberá almacenar y cómo será almacenada para proveer eficiencia en cuanto, a respuesta al pedido de información de los usuarios, flexibilidad y extensibilidad.

El papel que cumple el perfil en estas aplicaciones es importantísimo, ya que cuando se deben personalizar los aspectos de la aplicación se interactúa con el perfil del usuario para lograr las adaptaciones correctas basándose en los gustos e intereses del mismo. El perfil debe ser capaz de almacenar toda la información necesaria para personalizar cada uno de los aspectos personalizables de la aplicación, poder almacenar las preferencias, intereses, características, y asimismo ser capaz de guardar información propia de cada uno de los usuarios como puede ser el nombre, apellido y localidad, entre otros. Almacenar y mantener toda esta información no es una tarea fácil, referente a la manipulación y recolección de los datos. Además, toda esta información no solo es necesaria para la personalización, sino que es de muchísima importancia en las aplicaciones de comercio electrónico, en donde ofrecerle productos o servicios interesantes para los usuarios producen como consecuencia grandes ganancias para los propietarios de la aplicación.

Como conclusión de esta sección, se puede afirmar que las aplicaciones personalizadas sin almacenamiento de los perfiles de usuario son imposibles de realizar, ya que cumplen un rol importantísimo que es el de interactuar con el corazón de la aplicación para realizar la tarea de personalización.

2.4.1.¿Qué representa un Perfil de Usuario?

Un perfil de usuario representa la información que posee la aplicación sobre cada usuario, asimismo, se debe destacar una particularidad importante para tener una mayor visión de lo que puede llegar a representar. Cuando se trata de aplicaciones personalizadas no siempre se está haciendo referencia a aplicaciones que personalizan sus aspectos a usuarios individuales. Existen aplicaciones en las que cada usuario no es representado de manera individual, eventualmente son grupos de usuarios que contienen alguna característica en común y para la aplicación esas características los distinguen a unos de otros y la personalización se encuentra orientada a dichos grupo. Estos grupos por ejemplo, pueden ser diferentes categorías de usuarios en una aplicación de un banco, que pueden personalizarse orientándose a clientes de oros, clientes con mayor categoría en el banco, clientes de plata, clientes con categoría media, y clientes de bronce, en general clientes más recientes o con poco capital ingresado al banco. En este caso presentado, la personalización estaría apuntada a personalizar los grupos de categorías de clientes, ofreciéndoles mayores servicios a los clientes de oro, y quizás menos servicios o diferentes ofertas a los clientes de bronce. El planteo que surge en estos casos es cómo se almacena el perfil, ya que no se debe guardar información de un único usuario sino de un grupo de ellos. Se debe tener en cuenta, que en estas situaciones los clientes siguen siendo tratados de una manera individual, o sea, se almacenan sus datos personales entre otros, pero además una característica importantísima para la aplicación es a qué grupo pertenece.

Existen dos fuertes opciones, según se presenta en [19], se puede representar cada usuario como miembro de un grupo y crear agregaciones de perfiles de usuarios, o directamente mantener los perfiles de los usuarios individualmente y actualizarlos con cualquier cambio que se produzca sobre su grupo. Cuando a los usuarios se los ubica como un grupo, el método que podría ser utilizado es la creación de perfiles de usuarios agregados basados en reglas y patrones que son extraídos aplicando técnicas de Web mining. Aprovechando esta técnica u otras existentes las aplicaciones pueden ser apropiadamente personalizadas orientándolas a grupos.

2.4.2. Información de los Usuarios

Los perfiles de usuario proveen información sobre las personas que utilizan la aplicación. Se puede describir de una forma figurada que la información de los perfiles se compone de dos secciones [21]. Una de las secciones, contiene información personal del usuario, tal como nombre, apellido, edad, sexo, localidad, situación familiar, estado civil, si tiene hijos, trabajo, entre otros. Esta información que compone esta sección del perfil por las características que posee puede ser vista como estática dado que la información que contiene no cambia, o raramente es alterada. La otra sección, contiene información de los intereses y preferencias de los usuarios, adquirida con los formularios de la registración y cuestionarios, o inferida mientras los usuarios navegan por la aplicación. Esta sección puede ser como dinámica, ya que contiene información que varía de manera constante. Esta última información es la pieza imprescindible para posibilitar a la aplicación lograr una buena personalización, dado que esta sección almacena información sobre preferencias de la estructura, la interfaz, contenido, datos de hardware y otros datos necesarios para la adaptación de la aplicación a los usuarios. Algunos datos que componen esta información no son obtenidos de manera explícita, ya que no se le pregunta al usuario sobre sus preferencias sino que es la aplicación la encargada de observar el comportamiento del usuario y almacenar la información que es útil para esta tarea. El tema de obtención de datos de los usuarios será presentado con más detalle en la sección siguiente. La aplicación deberá conocer e ir filtrando qué información desea almacenar del usuario y cual no es necesaria. La Figura 2.3 muestra

una visión figurativa de las secciones que componen al perfil de usuario, basándose en la información que lo compone.

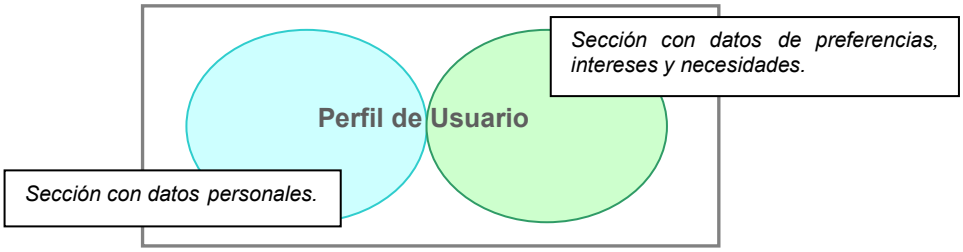

Figura 2.3: El Perfil de Usuario figurativamente se compone de dos secciones.

Una cuestión importantísima a destacar, es que el usuario nunca ve su perfil de usuario, y no es manejado por él, sino que es controlado y manipulado por la aplicación. Por este motivo, se debe proveer de alguna manera la posibilidad de que el usuario pueda cambiar la información de su perfil. Puede suceder el caso de que el usuario en un momento estuvo interesado por cierto tipo de información y este dato de interés haber sido almacenado como preferencia del mismo, y luego, pasado el tiempo ya no se interesa más por este gusto. Si esto ocurre y la aplicación no permite el cambio de perfil, el usuario va a ver páginas adecuadas a esta preferencia que no es más de su interés, lo que provocaría un mal resultado del proceso de personalización que sería solucionado si se permitiera manejar ciertos aspectos del perfil al usuario. Otro caso que puede ser posible, es en una aplicación como Amazon, un usuario compra un libro para una sobrina de poca edad y que este interés sea guardado como preferencia en el perfil del usuario, no siendo lógico ya que lo hizo una vez y es posible que no compre más libros para chicos de esa edad, por lo cual la aplicación, o debe ser suficientemente inteligente como para darse cuenta de no almacenar esas preferencias del usuario en su perfil, o darle la posibilidad al mismo de modificarlo y eliminar estos datos. Por dichas razones, se debería evaluar de brindarle a los usuarios la posibilidad de ver y modificar sus preferencias, así la aplicación contiene información que puede ser actualizada y modificada por el mismo usuario logrando así un buen nivel de personalización.

Al aparecer las aplicaciones personalizadas, y con éstas el almacenamiento de información que contiene el perfil de usuario que son datos personales más otros datos que proveen sus características, surge un tema muy importante, delicado y complicado, la privacidad y la seguridad de dicha información. En estas aplicaciones, el usuario deja de ser una persona anónima que accede a la aplicación y se convierte en un usuario con nombre, apellido y características de él. Este almacenamiento contiene información personal y muy propia de cada una de las personas que acceden a la aplicación, por lo cual, se deben proveer mecanismos de seguridad y tener conciencia de respetar la privacidad de la información de los usuarios, en otras palabras, tener absoluta reserva de ésta y no compartir o hacer negocios de venta de dicha información a otras compañías. Por esta razón, es necesario que la información de los usuarios sea guardada con absoluta reserva, empleando sistemas de seguridad para evitar que sea alcanzada por personas ajenas a su acceso. Se deberán utilizar estrategias de seguridad como claves de acceso, tener la información delicada en forma encriptada, y que sea almacenada en máquinas en donde se tengan acceso restringido, entre otras estrategias que pueden ser utilizadas. Algunas compañías que poseen estas aplicaciones, además, contratan empresas especializadas en seguridad para que verifiquen la seguridad sus sistemas de almacenamiento, y que también, sugieran cambios necesarios para mejorar sus

estrategias de seguridad[3]. De este modo, se minimizará el riesgo de la distribución de la información a manos de gente que no corresponda.

El tema de seguridad y privacidad es muy complejo, fue y sigue siendo muy debatido. Para resolverlo se dictaron leyes que protegen los datos de los usuarios, ver la sección 1.4.3 en donde se desarrolló con detalle este tema.

Para concluir con este tema, se presenta una solución muy conocida, P3P[4] (Platform for Privacy Preferences), una recomendación propuesta por W3C como se explica en [19], que nace ante la necesidad de garantizar la privacidad en la Web. Este lenguaje estándar permite a las aplicaciones Web realizar sus prácticas de privacidad en un formato estandarizado, que proporciona a los usuarios una forma sencilla y automatizada de controlar en mayor medida el uso que se hace de su información personal en los sitios Web que se visitan. P3P mejora el control del usuario al colocar políticas de privacidad donde los usuarios pueden encontrarlas, en un formato en el que los usuarios pueden entender y, lo más importante, con la posibilidad de que el usuario actúe sobre lo que ve. Permite a los sitios Web trasladar sus prácticas de privacidad a un formato estandarizado y procesable por dispositivos que puede ser recuperado de forma automática y que además puede ser interpretado fácilmente por los navegadores de los usuarios. En conclusión, P3P proporciona a los usuarios Web facilidad y regularidad a la hora de decidir si quieren o no, y bajo qué circunstancia, revelar información personal. La desventaja que se puede encontrar es que no provee un mecanismo para asegurar a los usuarios que la aplicación actúa de acuerdo a las políticas que son propuestas.

Como esta solución propuesta, existen varias más que deben ser tenidas en cuenta cuando se quiera tratar el tema de desarrollar aplicaciones personalizadas y se necesite evaluar el tema de la privacidad. Sí o sí, se debe asegurar a los usuarios que sus datos son almacenados y accedidos en forma segura, si esto no sucede, se logra que los usuarios que no confían en la aplicación y no hagan uso de la misma, perdiendo de esta manera gran cantidad de usuarios.

2.4.3. Obtención de la Información de Usuario

La recolección de las características de los usuarios tales como sus intereses, deseos, preferencias y necesidades, precisan ser capturadas y procesadas para luego ser almacenadas en los perfiles de usuarios. Esta obtención de información puede ser efectuada de dos maneras o realizando una combinación de dichas dos. Efectuándola de una manera explícita (o adaptable), en donde el usuario es el encargado de brindar su información a la aplicación, de una manera implícita (o adaptativa), en donde es la aplicación la que infiere la información sin que el usuario se de cuenta que esta tarea está siendo realizada, y la tercera es una combinación de ellas [22].

En la obtención explícita, son los usuarios los encargados de proveer sus datos e indicar sus preferencias para que sean almacenadas en sus perfiles. Los datos de los usuarios pueden ser llenados a través de formularios, o accediendo a páginas destinadas a establecer sus preferencias, como sucede en MyYahoo. En estos casos, los usuarios tienen el absoluto control de su información y pueden modificarla. Gran cantidad de la información obtenida explícitamente es adquirida en el momento de la registración. En la misma, se les pides a los usuarios, además de sus datos personales, llenar datos que se encuentran relacionados con sus intereses y preferencias, lo cual tiene como objetivo que el perfil de usuario contenga los datos que sean útiles para la realización de la personalización desde el momento en que el usuario comienza a ser parte de la aplicación. Este estilo de obtención de datos, requiere esfuerzo del usuario e inversión de

3 El sitio Yahho por ejemplo utiliza muchas de estas técnicas nombradas, para más información ver referencia [16].

4 Para obtener más información sobre P3P se puede acceder a http://www.w3c.org/P3P/.

tiempo en el llenado de sus datos y preferencias, dependiendo meramente de la disposición y habilidad de los usuarios para el llenado de la información. Al poseer formularios en páginas destinadas a este propósito, los usuarios luego de haber completado la tarea de proveer explícitamente sus datos, tienen la posibilidad de realizar cambios en sus preferencias o intereses según deseen o sea necesario, dado que en general, las aplicaciones que usan este tipo de obtención de datos brindan a los usuarios la posibilidad de modificar su perfil reutilizando estos formularios o páginas, y de este modo, logrando que su personalización se realizase con datos precisos y actualizados ya que son completados por los mismos usuarios y no existe posibilidad de error.

Si bien resulta una ventaja esta manera de obtención de datos, ya que si el usuario se toma el tiempo de completar correctamente sus datos y los mantiene actualizados, logrando que el perfil del usuario se encuentre completo y con información correcta, puede observarse una gran desventaja. Ésta se produce como consecuencia de que demanda cierto tiempo y un considerado trabajo por parte de los usuarios el llenado de información al ingresar sus datos, y luego, lo que resulta más problemático, es lograr que los usuarios efectúen la actualización de dichos datos. Sino se cuenta con usuarios interesados y dispuestos a tomarse este trabajo, la personalización no da buenos resultados, por la razón de que los datos que son almacenados son fundamentales para lograr una buena personalización y si estos no son llenados o no se toma conciencia que deben ser actualizados, los resultados son pobres y errados ocasionando que el usuario se sienta defraudado y no vuelva a ingresar a la aplicación ni la recomiende a otras personas que accedan a la misma.

Con respecto a los formularios a completar, se puede destacar que se debe tener mucho cuidado y precisión al armarlos puesto que deben ser llenados por los usuarios. Si los formularios son cortos, los usuarios los llenan con gran satisfacción, pero luego los resultados de la personalización son pobres como consecuencia de que se posee poca información de cada uno de ellos. En el caso de que los formularios sean muy extensos, los resultados de la personalización serán más precisos y seguramente provocarán que los usuarios se encuentren complacidos, pero son pocos los usuarios que se encuentran dispuestos a completar el extenso formulario presentado. Lo que generalmente sucede, es que al acceder a la aplicación y observar la extensión del formulario a completar abandonen el sitio, o dejen incompleto gran parte del mismo, y de esta manera no llegar a descubrir los beneficios de la aplicación. Como conclusión, los formularios deberán estar bien diseñados, logrando que sean de gran alcance en cuanto a la información a solicitar y que de alguna manera estén bien organizados para que sean de fácil comprensión e incentiven a los usuarios a completarlos.

La otra manera que existe para obtener datos es de forma implícita, la aplicación es la encargada de monitorear la actividad de los usuarios y recolectar sus datos, sin que éstos se enteren que esta tarea se está llevando a cabo[5]. Es preciso aclarar, que el usuario tiene conocimiento de que esta tarea está siendo realizada, dado que se deben cumplir las reglas de privacidad y el usuario debe estar informado de que la aplicación realiza algún tipo de acción para recolectar sus datos, pero el usuario no es conciente en el momento que esta recolección esta siendo realizada, ni nota ningún cambio como para advertir que esta tarea se está realizando. El recolectar información implícitamente implica que la aplicación controla y guarda los movimientos de los usuarios mientras que ellos interactúan y navegan dentro de la aplicación. Esta tarea consiste en guardar información sobre el comportamiento que los usuarios llevan dentro de la aplicación, consistiendo en

5 Existe una herramienta llamada ClixSmart, se basa en esta manera de obtener datos. Fue desarrollada por el departamento de Ciencias de Computación de la University Collage Dublín. Esta herramienta realiza dos tareas, la de monitorear la actividad online de los usuarios y automáticamente construye los perfiles de usuarios para capturar sus intereses y preferencias. Se puede obtener más información en [23].

datos que incluyen, los links visitados, de dónde viene el usuario navegando, la ruta que utilizó para navegar a través del sitio, categorías de productos seleccionados, tiempo utilizado dentro de una página o haciendo conexiones entre los links visitados y sus consecuencias, como pueden ser la situación de luego de visitar un link poner un producto en el carrito de compras, estas son algunas entre otras tantas opciones para obtener información de los usuarios. Esta información que se va recolectando se procesa y se almacena en los perfiles de los usuarios. Lo que sucede con dicha información es que los usuarios no tienen control sobre ella, y si quisieran efectuar algún cambio esto no es posible de realizar, la misma solo se actualiza con las nuevas interacciones que se efectúen. La desventaja que esto ocasiona es que la información obtenida por la aplicación podría no ser la correcta, ya que el usuario quizás pudo haber demostrado interés por alguna clase de producto, esta preferencia haber sido guarda, y que no corresponda a un gusto real del mismo. Este caso no podría tener lugar cuando la información es obtenida de manera explícita. Otra desventaja que presenta, es que algunos usuarios se sienten invadidos al no saber qué información precisa está siendo obtenida mientras ellos navegan, sintiendo que la aplicación se comporta de manera entrometida. Además, utilizando técnicas para observar el comportamiento de los usuarios se provoca una gran demanda de tráfico en la red, y se debe contar con redes rápidas para que no se encuentre afectada la performance de la aplicación. Una ventaja de este método es que los usuarios no tienen que llenar formularios con sus preferencias, sino que es la aplicación la encargada de realizarlo, y de este modo no implica ningún esfuerzo por parte de ellos. Como ventaja muy importante se debe destacar que observar el comportamiento de los usuarios es una manera ideal de conocer sus gustos y preferencias, por lo cual utilizar esta manera de obtención de datos provee a la aplicación de un gran conocimiento del usuario y provoca que la personalización sea de un alto nivel.

La complicación que se presenta al querer implementar esta manera, es que se debe estudiar y diseñar cuales van a ser las acciones que realicen los usuarios que van a ser consideradas como indicaciones de comportamiento de los mismos para que esta información sea almacenada en sus perfiles.

Como tercera opción, existen las aplicaciones que efectúan la obtención de datos utilizando las dos maneras expuestas anteriormente. Estas aplicaciones deberán presentar un formulario solamente con datos personales a llenar por los usuarios y luego estudiar el comportamiento y preferencias de los mismos mientras que navegan por el sitio, sin tener que pedirles que completen formularios extensos. Esta opción es la que produce mejores resultados, dado que obtiene información de los usuarios de manera explícita e implícita y de esta forma obtiene abundante conocimiento sobre ellos y cuanto más se los conoce más acertados serán los resultados de la personalización.

El objetivo de la personalización es simplemente entender los deseos y necesidades de los usuarios, para brindarles como resultado información interesante y hacerlo en un ambiente que sea cómodo y adaptado a ellos. La clave es entender a los usuarios y en términos de procesamiento de datos, organizar y construir estructuras de datos para almacenar sus datos, y recolectar dicha información en la interacción para lograr como efecto una buena interacción y personalización de ellos. Para lograr esto, los usuarios deben tener bien presente que cuanta más información le brinden a la aplicación mejor serán los resultados que se obtengan de la personalización. Y en el caso de que se realice obtención implícita de los datos, cuanto mayor sea el tiempo que se dediquen a navegar por la aplicación, mayor será la obtención de datos que se puede recolectar sobre ellos. De esta manera, el conocimiento que se tiene sobre los usuarios es superior y los resultados de la personalización mejoran enormemente, ya que cuanto más se conocen sus preferencias y sus gustos, mejor se adaptan las aplicaciones a los mismos.

2.5 Reglas de Negocio

Básicamente las reglas de negocio son las reglas y estrategias implementadas en la aplicación con el objetivo de manejar y efectuar la lógica del negocio. Esta lógica contempla las políticas y estrategias propias del negocio [24]. Cuando se hace referencia a las reglas de "negocio", esto no implica que existen únicamente para las aplicaciones de comercio electrónico, sino también se relacionan a aplicaciones tales como bancos, manejo de alumnos de una facultad, entre otro millón de aplicaciones que pueden exigir de una lógica de utilización de reglas y restricciones. Por ejemplo, en los bancos estas reglas son las que regulan los descuentos o aumentos realizados a las cuentas, se pueden realizar descuentos en el pago de las cajas de ahorro si la persona tiene asegurada su casa a través del banco, en el caso de no poseer dicho seguro no se realiza el descuento. Como esta regla, existen muchísimas más, además van cambiando constantemente, se van agregando nuevas, o eliminando algunas que ya no se encuentran en vigencia dentro de las reglas del negocio. Se puede decir que en general, dependiendo de una condición dichas reglas toman decisiones al respecto, logrando con la implementación de dichas reglas soluciones a los problemas de tomar decisiones que dependen de variables y que aplican cierta lógica.

Las reglas de negocio son un tema sumamente importante para las aplicaciones de comercio electrónico. Como esta tesis se encuentra orientada a las aplicaciones Web personalizadas de comercio electrónico, no se podía dejar de tratar, por lo cual, se presentan algunas secciones que describen el tema explicando qué son y cómo las reglas de negocio afectan a las aplicaciones personalizadas de comercio electrónico. No se pretende profundizar sobre el tema ni tratar de resolverlo, ya que representa un asunto muy complejo. En el ámbito de la investigación, se ha estudiado e investigado como tema particular y si se trataba con detalle dentro de este trabajo se hacía muy extenso, además de no encontrarse relacionado con el objetivo del mismo. Por lo cual, solo se presenta la idea del tema y su explicación, y se deja abierto para que si el lector se encuentra interesado en el mismo busque más información sobre este tema.

2.5.1. ¿Qué representan las Reglas de Negocio?

Las reglas de negocio surgen a partir de que de alguna manera se deben representar en las aplicaciones las políticas del negocio o estrategias del mismo. Dichas reglas son parte de las prácticas y procedimientos de una organización, representan la sección especulativa y lógica del negocio. Por ejemplo se pueden desarrollar reglas relativas a descuentos y ofertas a realizar a los usuarios entre otras posibilidades. Estas reglas pueden cambiar de un momento a otro, ser agregadas o eliminadas, para adecuarse a los cambios del mundo de los negocios [25]. Estos cambios que suceden en el negocio de la aplicación, conducen a realizar cambios en el software que implementa a la misma proveyendo las funciones del negocio. Como se sabe, las aplicaciones Web para ser implementadas se encuentran compuestas por varias secciones que las componen, workflow, el manejo de interacciones entre el cliente y el servidor, la base de datos, las cuestiones de interfaz, entre otras. Por este motivo, si las reglas de negocio se encuentran implementadas sin realizar un diseño o sin tener bien claro la dificultad que implican y sin tener en cuenta las modificaciones constantes que sufren, y se implementan estas reglas como parte del código como sucede en algunas aplicaciones, realizándolas con *if then else* para tomar decisiones según una condición, o utilizando la sentencia *case*, provocará como consecuencia un imposible mantenimiento de la aplicación. Se deben modificar las reglas pero para lograrlo, si es que se puede realizar, se debe modificar mucho código, produciendo cambios costosos, consumiendo muchísimo tiempo y dedicación, además se debe tener en cuenta de no alterar otras reglas existentes. Considerando el número de veces que las reglas son cambiadas, los costos que este tipo de cambio produce y la complejidad que pueden llegar a representar implementarlas, se

ha estudiado y llegado a la conclusión, como se describe en [26], de que resulta más eficiente definir las reglas de negocio usando una ingeniería de reglas realizada aparte de la aplicación y luego integrarlas a ésta. De este modo, cuando las reglas deben ser modificadas, se admite que puedan ser cambiadas dentro de este pequeño módulo que las contiene e integrar estos cambios luego de realizados dentro de la aplicación. Considerando esta separación de tareas, existen herramientas que automatizan este tipo de reglas, dando a sus clientes la posibilidad de determinar sus reglas y poder mantenerlas con poco esfuerzo[6]. Estas herramientas son utilizadas en caso de que la complejidad de la aplicación así lo requiera, si las reglas de negocio son escasas y no son de gran complejidad, éstas pueden ser integradas en la aplicación dado que al ser pocas no se complica su mantenimiento, pero se debe tener en claro que en caso de gran incremento de las mismas se debe pensar en desacoplarlas.

Como este trabajo se encuentra orientado a las aplicaciones Web personalizadas de comercio electrónico no se podía dejar de tratar las reglas de negocio que forman una parte esencial de estas aplicaciones, dado que desarrollan la sección comercial y la lógica del negocio reflejando las políticas y estrategias propias de los negocios, pero no se centran estas secciones en resolver el problema sino se trata como un módulo a parte por la complejidad y diversidad de implementaciones que presenta este tema.

2.5.2. Aplicaciones de las Reglas de Negocio a la Personalización

En las secciones anteriores se explicaron las reglas de negocio que afectan a aplicaciones comunes, en esta sección se presenta como afectan a las aplicaciones personalizadas. En estas aplicaciones, las reglas de negocio se encuentran aplicadas a los diferentes usuarios que acceden a la aplicación, y quizás la cantidad y la complejidad de las mismas aumenten como consecuencia de que muchas reglas ahora deberán distinguir entre los diversos usuarios y sus diferentes preferencias y características. Algunas de las reglas se introducen con la propia personalización, dado que pueden existir reglas que se apliquen a algunos usuarios y a otros no. Estas reglas pueden afectar por ejemplo a las políticas que se le aplican a los carritos de compra, los usuarios pueden tener un descuento o pueden aplicarse reglas de negocio sobre los productos comprados pero que varíen las reglas según preferencias de compra o características de los usuarios, o quizás que ya se presente la forma de pago sabiendo por los antecedentes del usuario cual es su preferencia, lo que implica personalización de las reglas de negocio. Éstas pueden incluir personalización sobre la forma de pago, envíos de los productos, descuentos sobre los productos, entre otras. El usuario debe tener para estos casos el historial sobre sus compras, por lo cual las reglas deben realizar interacciones con los perfiles de los usuarios para obtener los resultados personalizados. En los casos de las aplicaciones no personalizadas las reglas solo se aplican sin depender de las características de los usuarios, lo cual produce que las reglas sean considerablemente menos complejas que en las aplicaciones personalizadas.

Se debe distinguir entre las reglas de negocio personalizadas que afectan a la aplicación y las acciones realizadas para personalizar la aplicación. Claramente, la diferencia es que las reglas de negocio contemplan reglas mismas del negocio, como son los descuentos, las formas de pagos, que por ejemplo no tienen nada que ver con las preferencias de interfaz del usuario que son acciones propias de la personalización. Se debe tener bien clara esta diferencia para no confundir una acción que forma parte de la sección de la aplicación con una regla de negocio ya que las reglas de negocio se implementan en un módulo separado.

---

6 Para los interesados en el tema de reglas de negocio, se puede observar en la referencia [26] una herramienta que provee manejo para implementar reglas de negocio.

Capítulo 3

Trabajos Relacionados

Este capítulo se encuentra destinado a exponer otros trabajos existentes que soportan aplicaciones personalizadas. Estos mismos fueron seleccionados entre varios otros por ser las que más se adecuan como aporte de alguna idea o innovación. Algunos de los trabajos que son expuestos realizan algún aporte que puede ser tomado en cuenta para desarrollar la arquitectura modular que se presentará en este trabajo, o proponen una idea nueva que puedan presentar un punto de vista muy diferente.

La sección 3.1 describe la motivación sobre la presentación de los trabajos que son expuestos en este capítulo. Las cuatro secciones siguientes corresponden cada una a un trabajo presentado, describiendo su funcionalidad, diseño y características. Dichos trabajos son presentados de una manera general, sin introducir detalles que no son de importancia y que no realizan ningún aporte a al desarrollo de la tesis.

El propósito de este capítulo es el de exponer otros trabajos que fueron desarrollados para resolver las aplicaciones Web personalizadas, y de esta manera, presentar otras percepciones y soluciones a un mismo problema. Éstas pueden ser o no resueltas de una misma manera, pueden existir mejores o peores soluciones, pueden poseer ventajas y desventajas, pero el objetivo es que contribuyan con ideas y colaboren en pensar de un modo más amplio con respecto a las soluciones existentes y que pueden ser tomadas en cuenta para solucionar algún aspecto de la arquitectura que se presentará en esta tesis.

3.1 Motivación

Considerando que el objetivo de este trabajo es el de desarrollar una arquitectura que soporte aplicaciones Web personalizadas se realizó un estudio de otros trabajos que fueron desarrollados con el mismo objetivo. El propósito de investigarlas fue el de obtener conceptos sobre otros diseños que fueron desarrollados y las contribuciones que los mismas realizaron, qué aspectos de la personalización soportan y cómo logran resolverlo. Gran cantidad de los trabajos encontrados, solo abarcan algunos aspectos de la personalización, y en algunos casos, aspectos que no son relevantes para este trabajo, tales como arquitecturas que tienen como objetivo la búsqueda personalizada de documentos en la Web, por lo cual no fueron incluidas en este trabajo. Asimismo, no se han encontrado muchos trabajos que ciertamente realicen el enfoque de la personalización como se verá en este trabajo, por este motivo, se realizó una selección entre los hallados los cuales contienen algún aporte, una buena idea, una visión diferente a la arquitectura expuesta en este trabajo o que resultan interesantes y se tomó la decisión de incluirlas.

A continuación se presentan en cada sección cada uno de los trabajos que fueron investigados. Se realizará un resumen de cada una de ellos, conteniendo sus principales características sin presentarse los detalles que son irrelevantes y no ofrecen ningún aporte a esta investigación.

Un punto a aclarar de este capítulo es que los gráficos expuestos son copias tomadas de los trabajos originales, no se rehicieron nuevos gráficos dado que querían mantenerse las ideas que cada autor quiso expresar. Por esta razón sucede que en alguno de ellos su texto no sea muy legible.

3.2 Mediator que comunica Usuarios con Servicios Personalizados

Los proveedores de los servicios de personalización son lo encargados de desarrollar aplicaciones Web que contemplen la personalización. La utilización de la personalización en diferentes aplicaciones Web propone a cada usuario ir navegando de aplicación en aplicación personalizando en cada una de ellas sus características que quedarán guardadas en los perfiles. La idea de esta arquitectura es evitar este trabajo que debe realizar cada usuario, evaluando para lograr esto las necesidades de ellos y los proveedores de servicios. Se propone una arquitectura modular para soportar servicios personalizados [27] que integran características de diferentes fuentes bajo un mismo *mediador* y de esta manera lograr satisfacer a los usuarios y que los proveedores de los servicios obtengan ventajas.

Se propone una arquitectura flexible que posee un mediador (*Personalization Mediator)* que actúa como un intermediario de servicios de personalización, entre los proveedores de personalización y los usuarios. En esta arquitectura los proveedores de los contenidos personalizados publican sus servicios de personalización a través de una interfaz unificada de este mediador. Un usuario provee su información del perfil al mediator que se encargará de encontrar y llamar al servicio que mejor se adecue al pedido de éste. Estos servicios finalmente retornan el mejor contenido personalizado adecuado al usuario que lo solicitó.

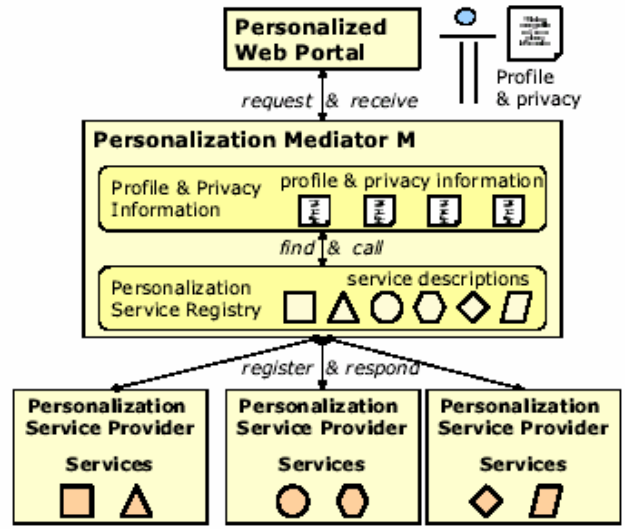

Figura 3.1: Arquitectura que utiliza un mediator para obtener personalización.

Como se ilustra en la Figura 3.1, en la arquitectura propuesta los proveedores de servicios de personalización (*Personalization Service Provider*) ofrecen sus facultades de personalización. Los servicios de personalización de los diferentes proveedores son registrados en el mediator el cual almacena la descripción de dichos servicios en un registro de servicios de personalización. A través de un sitio Web (*Personalized Web Portal)* los usuarios realizan la solicitud de pedido de los contenidos personalizados. Este sitio le proporciona al mediador la información del usuario, tales como lugar que habita, intereses generales, entre otras. Todos estos pedidos y las transferencias de datos se encuentran realizados bajo políticas de privacidad propuestas por el mediator. Basado en la información de los usuarios y la privacidad de la información el mediator busca dentro de su registro de servicios de personalización el servicio más conveniente para el pedido del usuario. Una vez que el servicio es encontrado el mediador es el encargado de enviar la información del perfil del usuario al servicio. Éste es llamado y responde con el contenido personalizado el cual es finalmente recibido por el usuario.

El tipo de resultado no se encuentra especificado, el manejo de los resultados se dejan como responsabilidad a la aplicación del usuario. Esta ventaja produce que los servicios de personalización no deban tener que enviar los datos a una página Web, sino que se envían de un modo genérico y es la aplicación la encargada de adaptarlos. Estas interacciones se pueden observar en la Figura 3.2.

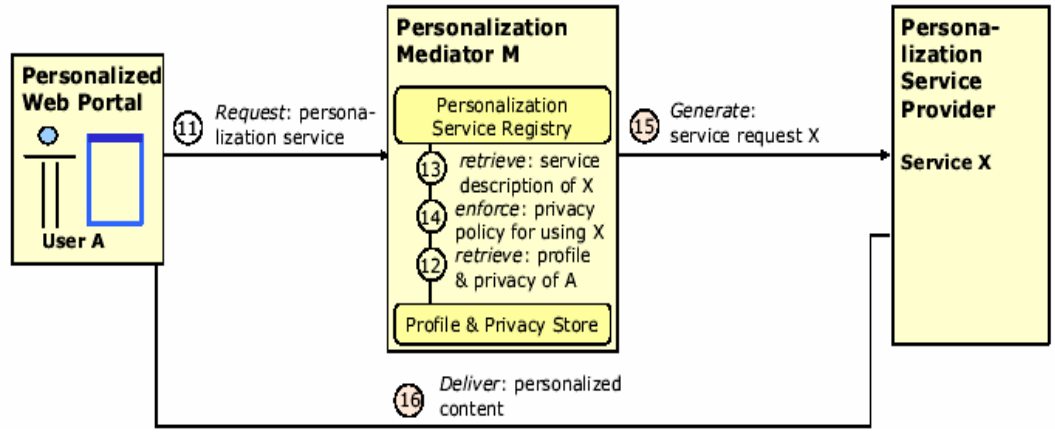

Figura 3.2: Interacción entre las partes de la Arquitectura.

El beneficio de la arquitectura propuesta es que los usuarios obtengan de una manera más fácil y unificada acceso a una ilimitada colección de servicios de personalización a través del mediador. Para los proveedores, la ventaja es que diferentes sitios con diferentes características a personalizar puedan embeber los servicios de personalización en una infraestructura genérica para aplicaciones personalizadas, como por ejemplo aplicaciones Web personalizadas.

3.3 Utilización de Smart Cards como Almacenamiento

Las ubicaciones más comunes para almacenar y manipular los perfiles de usuario son tres:

Localmente, en el dispositivo de cada usuario. El perfil sería almacenado en el disco rígido, donde él mismo tiene acceso y manejo de dicha información. Con este almacenamiento se producen ciertos riesgos. Si no se tiene precaución se podrían producir riesgos de seguridad, la información personal podría ser copiada sin autorización o modificada. Además, esta información se encuentra físicamente en un solo dispositivo, esto reduce a que la aplicación solo sea personalizada en ese contexto, sin permitir que se personalicen otras aplicaciones.

En el servidor que provee el servicio, siendo el mismo el encargado de mantenerlo y manipularlo. En este caso, la utilización de los perfiles se encuentra restringida a los servicios que brinda la aplicación, sin poder ser utilizada por otra.

La tercera opción, es ubicarlo en un servidor independiente. La aplicación interactuará con este servidor para lograr la personalización. Se realiza un intercambio entre los usuarios y el servidor de la aplicación, introduciéndose una tercera parte que es el servidor que mantiene los perfiles de los cuales depende la personalización. Esto introduce un tiempo más de respuesta a la aplicación para lograr la personalización.

El objetivo de este framework [28] es lograr personalizar para diferentes dispositivos, y que se pueda compartir el perfil con más de una aplicación, de este modo, otras aplicaciones pueden utilizarlo y la primera vez que el usuario ingrese a la misma pueda inmediatamente ser beneficiado con la personalización. Con estos propósitos y observando estas distintas ubicaciones de almacenamiento cada una con sus ventajas y desventajas, este framework propone almacenar el *Personal Profile Engine* en Smart Cards[7]. Presentando a las Smart Cards como el medio ideal de almacenamiento y

7 Smart Card, son tarjetas que contienen información y son transportables. Éstas son embebidas en los microprocesadores que utilizan la información que ellas contienen.

procesamiento, éstas proveen la seguridad y cualidades requeridas para contemplar los requisitos de privacidad y adaptación. Comparada con las soluciones basadas en almacenar la información centralizada en un servidor remoto, las ventajas presentadas por las Smart Cards son, que evitan los problemas de fallas de acceso y retraso en los intercambios, los perfiles se encuentran disponibles en cualquier contexto, el usuario controla su perfil (el usuario elige si presentar o no su perfil y hasta decidir si quiere destruirlo), y como última ventaja, las Smart Cards acrecientan el nivel de protección de los perfiles, dado que la resistencia de las mismas en cuanto a propiedad física y lógica aseguran su confiabilidad, comparándola con un servidor que requiere de un mecanismo de alto costo para ser protegido.

Dentro de este framework la personalización está vista como dos flujos de información, uno de entrada impersonal (*impersonal incoming flow*) y otro salida personalizado (*personalizad outgoing flow*). El flujo de información de entrada son los datos que se ingresan a la aplicación sin personalizar y se logran como resultado la salida de los datos personalizados. La información entrante puede ser de una página Web o de alguna otra forma como un query, y de salida producirá la presentación que corresponda.

En orden de realizar su tarea, el proceso de personalización debe tener una descripción del flujo de información de entrada para lograr entender la semántica de la información que entra. Esta descripción es denominada *metadata.* En este framework se asume que esta metadata está disponible para la personalización y no se considera como es obtenida. Además de esta información el proceso de personalización, que depende del Personal Profile Engine, maneja información propia de los usuarios.

Existen dos técnicas principales para realizar la personalización, *Content-based filtering* y *Collaborative filtering*. Content-based filtering realiza la tarea de machear la información de entrada y su metadata con el perfil de usuario. Esto permite elegir una acción de personalización para la información de entrada, siempre que se tenga suficiente descripción de ésta, y que el perfil contenga información correspondiente a la metadata. La técnica Collaborative filtering se sustenta en recomendar y obtener información personalizada para un usuario basándose en las preferencias y gustos de un perfil de usuario con características similares. O sea, se basa en obtener resultados buscando en la similitud de gustos y comportamiento otros usuarios. Estas dos técnicas combinadas proveen mejores resultados que cada una utilizada individualmente.

Las soluciones propuestas existentes sobre estas técnicas son todas basadas en un servidor centralizado, en donde todos los perfiles son accedidos de manera simultánea. Este framework proporciona una solución a estas dos técnicas basándose en que los perfiles de usuarios se encuentran descentralizados, brindando además una flexibilidad adicional.

Considerando el diseño del framework, donde cada perfil es almacenado de manera independiente en Smart Cards, el conjunto completo de los perfiles de usuarios no puede ser accedido simultáneamente, el cual provoca que la técnica collaborative sea inaplicable. Para resolver esto se propone un proceso de dos pasos de filtrado: un collaborative filtering es primero ejecutado remotamente por un *Collaborative Personalization Engine* (almacenado en el propio servidor de la aplicación o por uno remoto), el cual mantiene modelos colaborativos del usuario basados en anónimas manifestaciones sobre las preferencias de los mismos. Para construir estos modelos de usuarios la información de comportamiento debe ser enviada del lado del usuario a el *Collaborative Personalization Engine*. Por otro lado, como los modelos colaborativos de los usuarios no tienen que ser compartidos por las demás aplicaciones, los perfiles de usuarios pueden ser específicos para un negocio particular. El resultado de este primer proceso se lo denomina *personalization data*, que provee una descripción de cuales usuarios están interesados en los datos. Estos datos son agregados al flujo de entrada. En el segundo paso, el *Personal Personalization Engine* ejecuta un content-base filtering, que machea el perfil del usuario con la metadata y los datos de la personalización que provienen del primer proceso.

En la Figura 3.3 se puede observar el framework presentado, los tres rectángulos corresponden a la ubicación de los tres procesos. El *Collaborative Profile Filtering* es ubicado en un servidor remoto, los Personalization Services son colocados en el dispositivo del usuario, y el *Personal Profile Engine* es embebido en una Smart Card.

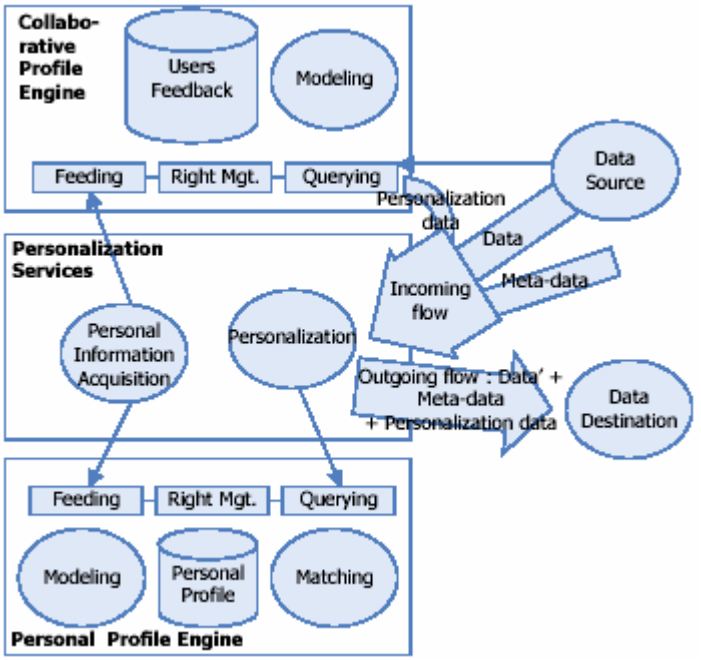

Figura 3.3: Framework de Personalización utilizando Smart Cards.

Se debe destacar que el *Personal Profile Engine* y el *Collaborative Profile Filtering* se encuentran totalmente independientes de la aplicación, la aplicación de dominio y el dispositivo. Por este motivo, pueden ser compartidos entre aplicaciones, y usados desde múltiples clases de dispositivos (teléfono celular, TV, PC, entre otros), a través de un operador.

Un aspecto clave de este framework es que separa los aspectos genéricos de personalización de la aplicación y los procesos específicos del dominio, logrando que sea reusable en diferentes aplicaciones, dispositivos y dominios.

3.4 Utilización de un Proxy como Almacenamiento

Esta arquitectura [29] propuesta provee soporte para aplicaciones Web personalizadas. Contiene un agente para manejar perfiles de usuarios (user profile management agent) que realiza la tarea de manejar los perfiles de usuarios y facilita el uso de técnicas de personalización avanzadas, tales como Collaborative filtering. Este agente de perfiles se encuentra ubicado en un proxy Web[8]. Otra de las posibilidades que se propuso fue que sea almacenado en un servidor separado, sin embargo, esta idea fue descartada dado que la principal e importante desventaja que posee es que se requiere una conexión de red en cada momento que la aplicación necesite un servicio y esto significa que se agrega un tiempo de espera. Surgió entonces, la idea de ubicar el agente en un proxy que fue basada en que hoy en día son muchas compañías utilizan uno o más

8 Actúa como intermediario entre el programa cliente y el servidor de información al que se quiere acceder. Éste es invisible para los usuarios, y actúa como un firewall, prohibiendo la entrada de visitantes no autorizados y el acceso desde la intranet a sitios Web no permitidos.

proxies entre sus intranets e Internet, como se muestra en la Figura 3.4, el User profile management Proxy. El proxy debe manejar, almacenar y retornar los perfiles de usuarios y es el mediador de las comunicaciones entre los usuarios y los servidores Web.

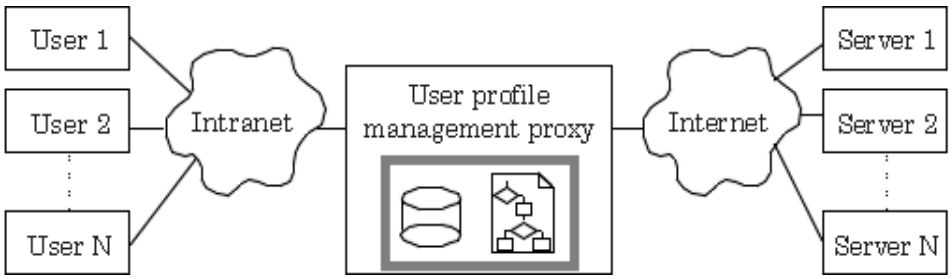

Figura 3.4: Agente almacenado en el Proxy.

Las ventajas que presenta esta arquitectura son:

- Es posible construir un agente que provee soporte para content-based filtering, para usuarios que comparten los perfiles, y Collaborative filtering entre los usuarios que acceden a Internet a través del mismo proxy.
- Reduce el tiempo de espera de los servicios. El agente de manejo de perfiles de usuarios actúa como un traductor, por lo tanto no requiere de una conexión extra para transferir el perfil del usuario entre el agente y la aplicación personalizada.
- Los proveedores del servicio se liberan de realizar la tarea de almacenar y administrar los perfiles de usuario. Esto simplifica y estandariza el desarrollo de las aplicaciones Web personalizadas y transfiere el costo del almacenamiento al lado del cliente.
- Permite al usuario acceder a aplicaciones Web personalizadas desde diferentes computadoras mientras que el acceso sea realizado a través del mismo Proxy.
- Es completamente transparente al browser, y por lo tanto, es compatible con los browsers comunes.

La utilización del User profile management proxy no es restringido a las compañías que poseen un firewall; un proveedor Web puede utilizarlo para centralizar la autenticación de sus clientes, para almacenar los perfiles de usuarios, y compartirlos entre los servicios que éste provee.

La funcionalidad es realizada de la siguiente manera, cada vez que el Proxy recibe una petición HTTP, chequea si la URL y el nombre de usuario se encuentran registrados. Si el correspondiente perfil de usuario existe, el perfil de usuario es obtenido de la base de datos y agregado a la petición HTTP, la cual es reenviada al servidor Web. La aplicación utiliza el perfil del usuario para sus propósitos de personalización, y si sucede que el perfil necesita ser actualizado, se incluye un nuevo perfil dentro de la respuesta HTTP. Como esta respuesta pasa a través del Proxy, el perfil es extraído, almacenado dentro de la base de datos y eliminado de la respuesta HTTP. Esto asegura una completa transparencia con respecto al browser.

Se logra proporcionar con esta arquitectura técnicas avanzadas útiles en la personalización, tales como, content-based filtering, de Collaborative filtering y la utilización de los perfiles compartidos para que puedan ser utilizados por varias aplicaciones. La técnica Collaborative filtering es provista por el manejo individual de los perfiles de usuarios que contribuyen al grupo. Se le da el nombre de group profile component dentro de la arquitectura. El almacenamiento de un componente de perfil de grupo es similar a almacenar un perfil de usuario regular, pero la recuperación del mismo es diferente porque el user profile management agent debe transferir a la aplicación Web todo los componentes de perfiles del grupo requerido. Para proveer la utilización de perfiles de usuarios que puedan ser compartidos, otra de las técnicas utilizadas, el user

profile management agent genera un catálogo que enumera los perfiles que son compartidos. Cada entrada del catálogo contiene no solo el nombre de un perfil de usuario y su creador (nombre de la aplicación, URL, entre otras), sino también la descripción de los contenidos de los perfiles de usuarios.

Como conclusión cabe destacar, que fueron realizaros dos prototipos de esta arquitectura los cuales resultaron efectuados con éxito. Además se evaluó la performance que poseían al realizar ejecuciones de cada una de las técnicas, y se observó un pequeño retraso en servicio de compartir los perfiles, que en un trabajo futuro se propone ser mejorado.

3.5 Recursos compartidos entre Usuarios con Intereses Similares

Esta arquitectura [30] tiene un nivel más abstracto que las vistas anteriormente, se encuentra desarrollada desde un punto de vista más conceptual. La idea que se propone es utilizar la tecnología P2P[9] para desarrollar una arquitectura Web personalizada que acceda a recursos estructurados de Internet y conocimiento compartido por miembros de grupos con intereses similares. Esta arquitectura usa una ontología basada en la representación de los recursos que admite una semántica de los recursos que son descubiertos y accedidos, y que reflejan los intereses de los usuarios.

Como idea principal de esta arquitectura se propone el concepto de una Personalized Web (PW), que representa un conjunto de vistas personalizadas de los usuarios de Internet, donde cada vista es responsable de especificar la necesidad personal o interés del usuario.

Conceptualmente una vista representa un sitio Web de recursos de Internet asociados con un contenido ontológico específico que refleja un interés específico del usuario. El mecanismo usado para crear una vista PW gira alrededor del concepto de divulgación activa y descubrimiento de los recursos, por lo cual, un agente de semántica referido como *Semlet*, actúa como un Proxy para un recurso específico y usa metadata asociada con el recurso, junto con una similitud métrica del usuario definido, para identificar y reunir grupos con intereses similares. El descubrimiento de nuevos recursos, acoplado con el desarrollo de nuevas vistas, proporciona el desarrollo de PW dinámicamente para incluir otras PWs, y de este modo, permite la formación "natural" de los grupos de usuarios de intereses similares.

Para soportar las funcionalidades de una PW, se requiere del diseño para cubrir la arquitectura básica del PW y del diseño de los protocolos y mecanismos requeridos.

Se encuentra formada por framework PW, en donde cada recurso es descripto por metadata, la cual puede ser vista como una definición del perfil personalizado del recurso. La metadata contiene por lo menos un atributo semántico el cual puede ser mapeado a una referencia semántica en la red ontológica. Los protocolos y mecanismos que colaboran con el PW deberían ser desarrollados por:

- La capacidad de descubrir recursos, basados en los intereses de los perfiles especificados por los usuarios, realizándolo de una manera transparente para los usuarios.
- La capacidad de formación de grupos con intereses similares, realizándolo de una manera transparente y dinámica.
- La automática divulgación de los recursos de los usuarios a los grupos formados que tienen un interés similar.

9 P2P (Peer-to-Peer), es un modelo de comunicación en el cual cada participante tiene las mismas capacidades y responsabilidades. En Internet, P2P es un tipo de red que le permite a un grupo de usuarios con el mismo programa de redes, conectarse entre sí e intercambiar archivos.

- Manejar la PW de los usuarios para dinámicamente capturar evolución de los intereses de los usuarios como los cambios de ambientes.

Para soportar estas funcionalidades, se propone una arquitectura PW, la cual se encuentra influenciada por las tecnologías, P2P, tecnología de agentes, y semantic Web. La tecnología P2P [31] ofrece la capacidad de organizar los recursos de Internet dentro de una colección de redes lógicas. Cada red puede ser vista como una comunidad de peers cuyos miembros comparten intereses comunes, prácticas y objetivos.

El objetivo del diseño básico de la arquitectura es proveer a los usuarios con la habilidad para adaptadamente crear sus propias vistas de Internet para encontrar sus intereses personales de una manera efectiva y escalable. La arquitectura básica se muestra en la Figura 3.5, en donde se observa que la misma comprende tres capas, llamadas PW Agent Layer, Ontology Overlay Network (OON) y la capa Internet Resource (IR).

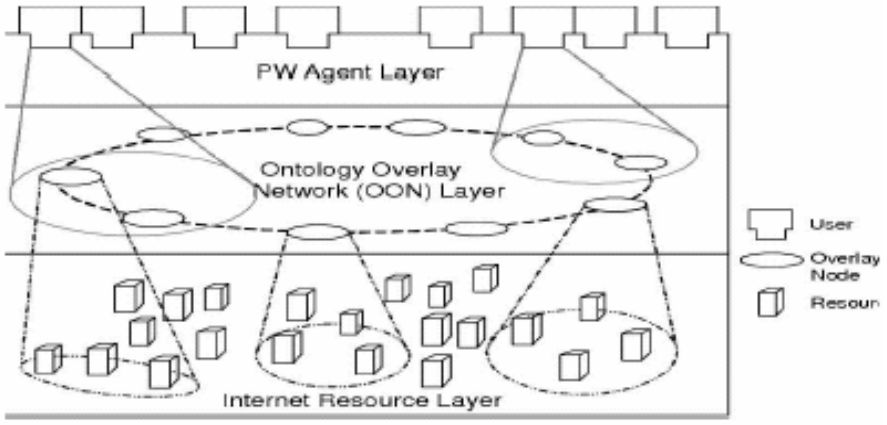

Figura 3.5: Las tres capas que componen la arquitectura.

Las dos primeras capas dependen de la capa IR para físicamente ubicar y acceder a los recursos. La capa IR utiliza corrientemente las capacidades de Internet para proveer estas funcionalidades.

La PW debe reflejar con el mayor detalle posible los intereses del usuario y permitir a la estructura evolucionar dinámicamente y realizarlo de una manera transparente para el usuario. Para lograr esto, la arquitectura propuesta emplea un acercamiento basado en un agente que comprende las funcionalidades requeridas para permitir a los usuarios expresar las vistas personalizadas de Internet, comunicando sus recursos a los grupos de intereses similares, y descubrir nuevos recursos de interés basándose en las similitudes de sus perfiles personalizados. En esto consiste la capa PW Agent Layer. El principio básico de esta capa es el agente Semlet, que cada uno se encarga de manejar los recursos de interés en el contexto de asociar perfiles de usuarios.

La capa Ontology Overlay Network, se utiliza como base para la representación de recursos, y permitir el desarrollo de indexación de recursos de forma eficiente. En este contexto, una ontología es definida como una especificación formal y explícita de una conceptualización compartida. Esta capa se encuentra estructurada por una tabla de hash distribuida (DHT) basada en P2P que integra un árbol Wordnet de ontología dentro de su estructura. La DHT es usada para deducir una estructura que regula la ubicación y acceso para un distribuido conjunto de recursos.

Además de estas capas, la arquitectura contiene mecanismos y estructuras de datos para lograr soportar las principales funcionalidades de una PW, las cuales utilizan algoritmos y protocolos.

Los principales componentes de esta arquitectura de Web personalizada son la ontología Wornet, la tabla hash distribuida utilizando P2P que cubre la infraestructura

OON, y por último el agente personalizado Semlet. Cada uno de estos componentes es muy importante para el desarrollo y funcionamiento de esta arquitectura. Se quiere lograr con la misma un alto nivel de escalabilidad con gran carga de datos, robustez y eficiencia.

Capítulo 4

Metodología y Patrones Utilizados

Este capítulo presenta una metodología y los patrones que fueron utilizados o colaboraron en la arquitectura que se expone en este trabajo. En este capítulo se realizará una descripción de cada uno de ellos con sus características, los conceptos que proveen y aportan; y por último, se dedica una sección que presenta los aportes que realizan a la arquitectura.

En primer lugar se encuentra la sección 4.1, la cual presenta la metodología y los patrones que serán desarrollados, brindando una breve explicación de cada uno de ellos. En la sección 4.2 se describe la metodología OOHDM exponiendo sus características y aspectos más importantes. Además, se realiza una exposición de las cuatro etapas que forman parte de esta metodología, describiendo en breves palabras cada una de ellas. A continuación, la sección 4.3, introduce el patrón MVC, realizando una explicación del mismo, su funcionalidad y exponiendo sus ventajas y características al ser utilizado en las aplicaciones que se desarrollan aplicándolo. La sección 4.4, presenta patrones utilizados en las aplicaciones Web personalizadas. Son cuatro patrones, Link Personalizado, Contenido Personalizado, Estructura Personalizada y Client-side Personalizado, se brinda de cada uno de ellos su descripción, objetivo y además se presenta un ejemplo de cada uno para un mejor entendimiento de los mismos.

Una vez desarrollados las secciones concernientes a la metodología y patrones, se encuentra la sección 4.5 cuyo propósito es el de explicar y presentar cada una de las características o aportes que realizaron la metodología OOHDM, el patrón MVC y los patrones de personalización Web al desarrollo de la arquitectura propuesta.

El objetivo de este capítulo es dar en conocimiento la metodología y patrones que realizaron aportes e influyeron en el desarrollo de la arquitectura propuesta, para que luego en los próximos capítulos cuando se visualice la arquitectura propuesta, se posean todas las bases para un entendimiento correcto y más sencillo de la misma.

4.1 Metodologías examinadas

Al desarrollar aplicaciones Web personalizadas se deben tener en cuenta metodologías, conceptos de diseño y tecnologías con las cuales se logre crear el dinamismo necesario que debe poseer una aplicación personalizada. Tanto para los aspectos de diseño como para el desarrollo de la aplicación, deben seleccionarse metodologías, conceptos de diseño y tecnologías con las que se logre alcanzar que las aplicaciones puedan ser extendidas o modificadas de manera sencilla, y además, que el impacto que provoque la realización de alguna de estas dos tareas sea mínimo o posible de contemplar y llevar a cabo sin tener que efectuar grandes cambios en la aplicación.

Para realizar la elección de las metodologías o conceptos a ser incorporados, que realicen aportes o proporcionen ideas a la arquitectura que se deseaba desarrollar, se investigaron cuales de las metodologías y técnicas de diseño existentes eran las que poseían las características necesarias para este tipo de desarrollo. Luego de un tiempo de investigación de características y diferencias entre las distintas metodologías y conceptos encontrados, fueron seleccionados por sus ventajas de independencia en el desarrollo y distribución de responsabilidades, la metodología OOHDM[10] [32] (*Método de Diseño Hipermedia Orientado a Objetos*) y el patrón de diseño MVC[11] [33] (*Modelo Vista Controlador).* Además de exponer OOHDM y MVC que realizan aportes a la arquitectura, se presentan en este capítulo cuatro patrones que son utilizados en las aplicaciones Web personalizadas. Esta presentación se realiza con el objetivo de observar claramente los patrones existentes específicos de las aplicaciones personalizadas y que la arquitectura debería contemplar para realizar el soporte de los mismos.

OOHDM es una metodología para el diseño de aplicaciones hipermediales que utiliza programación orientada a objetos, es una de las más conocidas hoy en día, y provee el dinamismo necesario tanto para el diseño de la aplicación como para mantenimiento de la misma. Por otro lado, MVC es un patrón arquitectural muy utilizado en las aplicaciones de escritorio que luego fue extendido para emplear en las aplicaciones Web, aprovechando sus ventajas de distribución de responsabilidades y desarrollo independiente para cada uno de sus módulos. Finalmente, los cuatro patrones de las aplicaciones personalizadas, Link Personalizado, Contenido Personalizado, Estructura Personalizada y Client-side Personalizado [34].

Con la metodología OOHDM y el patrón MVC, que proveen la capacidad de separar y realizar por módulos o capas las tareas que deben ser llevadas a cabo, agregando la particularidad de que poseen características para desarrollar el dinamismo que requieren las aplicaciones personalizadas, se logra consumar el objetivo propuesto de desarrollar una arquitectura modular. El objetivo no es el de separar la arquitectura en capas sin propósito alguno, sino con el objetivo principal y real de crear independencia en el desarrollo y mantenimiento entre los módulos propuestos, obteniendo de este modo la independencia que se pretende lograr en esta arquitectura, y además, aportando un buen desarrollo y un desempeño relativamente simple de la arquitectura.

La presentación de los cuatro patrones específicos de las aplicaciones Web personalizadas, se realiza con el propósito de exponer patrones que se presentan comúnmente en estas aplicaciones y que fueron tenidos en cuenta para el desarrollo de la arquitectura propuesta. De este modo, se logró que la arquitectura que se propone tenga soporte para poder implementar estos patrones tan importantes que se encuentran en las aplicaciones Web personalizaciones.

---

[10] Object Oriented Hypermedia Design Method.

11 Model View Controller.

4.2 Metodología OOHDM

OOHDM [35] es una metodología que tiene como objetivo proveer un conjunto de modelos de diseño y guías, para diseñar y construir aplicaciones hipermediales de alta calidad utilizando los conceptos de la programación orientada a objetos [36]. Dicha metodología, ha sido utilizada para diseñar diferentes tipos de aplicaciones hipermediales, como por ejemplo galerías interactivas, presentaciones multimedia y en aplicaciones Web, lo que atañe a este trabajo.

Una de las ventajas de esta metodología, comparada con otras, es que cuando realiza el desarrollo del diseño de la aplicación hipermedia centra especial atención en la navegación que contendrá la misma. Esto se realiza basándose en la teoría de que un buen diseño de navegación es la clave para que el usuario de la misma no se encuentre perdido en ningún momento, y además que el mismo pueda encontrar fácilmente la información buscada. Para la realización de este propósito OOHDM maneja conceptos como nodo, links, anchor y nodos en contextos entre otros, todos estos referentes a la navegación.

No obstante, el éxito de esta metodología es la clara identificación de tres diferentes niveles de diseño en forma independiente de la implementación, proponiendo por este motivo, que el desarrollo de las aplicaciones hipermediales sea realizado a través de un proceso compuesto por cuatro etapas. Estas cuatro son, la etapa de Diseño del Modelo Conceptual, Diseño del Modelo Navegacional, Diseño de las Interfaces Abstractas, y por último la etapa de Implementación. Con estas tres primeras se obtiene un conjunto de modelos orientados a objetos que describen cada uno de los modelos de la aplicación. Estos mismos son desarrollados en forma incremental, en cada paso cada modelo es construido y enriquecido. El objetivo principal de esta separación es la división de los modelos basándose en las responsabilidades dentro de una aplicación hipermedial, logrando de este modo la ventaja de la independencia en la realización de cada uno de ellos [37].

A continuación se presenta una breve descripción de cada una de las etapas que se emplean en esta metodología.

4.2.1. Diseño Conceptual

Durante esta actividad, se construye un esquema conceptual representado por los objetos del dominio, las relaciones y colaboraciones existentes establecidas entre ellos. Como se dijo anteriormente OOHDM es una metodología que utiliza la programación orientada a objetos, por lo cual, el esquema conceptual está construido por clases con métodos, relaciones entre ellas y subsistemas.

Este diseño conceptual es la realización de la construcción del diseño en objetos del dominio de la aplicación. Es realizado utilizando los métodos, patrones y técnicas de la programación orientada a objetos.

Para el desarrollo de esta diseño se utiliza notación similar a UML (Lenguaje de Modelado Unificado[12]) y tarjetas de clases y relaciones similares a las tarjetas CRC (Clase Responsabilidad Colaboración[13]). El esquema de las clases consiste en un conjunto de clases conectadas por relaciones. Los objetos son instancias de las clases. Las clases serán usadas durante el diseño de navegación para derivar nodos, y las relaciones que serán usadas para derivar enlaces.

12 Unified Modeling Language.
13 Class Responsibility Collaboration.

4.2.2. Diseño Navegacional

La primera generación de aplicaciones Web fue pensada para realizar navegación a través del espacio de información, utilizando un simple modelo de datos de hipermedia. Con el paso del tiempo las aplicaciones Web empezaron a progresar, con más variedades en el diseño y en la navegación; por lo cual se volvieron más complejas.

En OOHDM, la navegación es considerada un paso crítico en el diseño de este tipo de aplicaciones. Un modelo navegacional es construido como una vista sobre un diseño conceptual, refiriéndose a vista como visión de navegar los objetos propios de la aplicación.

El diseño de navegación es expresado en dos esquemas, el esquema de clases navegacionales y el esquema de contextos navegacionales. En OOHDM existe un conjunto de tipos predefinidos de clases navegacionales: nodos, enlaces, y estructuras de acceso. La semántica de los nodos y los enlaces son las tradicionales de las aplicaciones hipermediales, y las estructuras de acceso, tales como índices o recorridos guiados, representan los posibles caminos de acceso a los nodos.

La principal estructura primitiva del espacio navegacional es la noción de contexto navegacional. Un contexto navegacional es un conjunto de nodos, enlaces, clases de contextos, y otros contextos navegacionales (contextos que tienen la cualidad de ser anidados). Pueden ser definidos por comprensión o extensión, o por enumeración de sus miembros. Los contextos navegacionales juegan un rol similar a las colecciones y fueron inspirados sobre el concepto de contextos anidados. Organizan el espacio navegacional en conjuntos convenientes de nodos que pueden ser recorridos en un orden particular y que deberían ser definidos como caminos para ayudar al usuario a lograr la tarea deseada.

Los nodos son enriquecidos con un conjunto de clases especiales que permiten de un nodo observar y presentar atributos (incluidos las anclas, *anchor*), así como métodos (comportamiento) cuando se navega en un particular contexto. Cuando se navega por un nodo navegacional no se muestra toda la información de éste. En algunos contextos se podría decir que existen "diferentes formas" de mirar al un mismo nodo, presentando para los diferentes contextos, distinta información de un mismo nodo.

4.2.3. Diseño de Interfaz Abstracta

Una vez que las estructuras navegacionales son definidas, se deben especificar los aspectos de interfaz. Esto significa definir la forma en la cual los objetos navegacionales pueden aparecer, cómo los objetos de interfaz activarán la navegación y las otras funcionalidades de la aplicación, y qué transformaciones de la interfaz son pertinentes y cuando es necesario realizarlas.

Una clara separación entre partes, diseño navegacional y diseño de interfaz abstracta, permite construir diferentes interfaces para el mismo modelo navegacional, dejando un alto grado de independencia de la tecnología de interfaz de usuario.

El aspecto de la interfaz del usuario de aplicaciones interactivas (en particular las aplicaciones Web) es un punto crítico en el desarrollo, y las metodologías modernas tienden a descuidar este aspecto. En OOHDM, se utiliza el diseño de interfaz abstracta para describir la interfaz del usuario de la aplicación de hipermedia.

El modelo de interfaz ADVs (Vista de Datos Abstracta[14]) [38] especifica la organización y comportamiento de la interfaz, pero la apariencia física real o de los atributos, y la disposición de las propiedades de las ADVs en la pantalla real son hechas en la fase de implementación.

---

[14] *Abstract Data View*

4.2.4. Implementación

En esta fase, se deben implementar los diseños realizados en las etapas anteriores. Hasta esta etapa, todos los modelos fueron construidos en forma independiente de la plataforma de implementación; en esta etapa es tenido en cuenta el entorno particular en el cual se va a correr la aplicación y se deben tomar decisiones con respecto a las tecnologías que serán utilizadas. Se deberá examinar con atención como llevar los diseños realizados en OOHDM a las tecnologías que se utilizarán para la implementación, evaluando cuales son las más adecuadas y eficientes.

Al llegar a esta fase, se deben definir los ítems de información que son parte del problema del dominio. Se debe identificar también, cómo son organizados los ítems de acuerdo con el perfil del usuario y tareas; decidir qué interfaz debería ver, y cómo debería comportarse. A fin de implementar todo en un entorno Web, se debe decidir además qué información, tanto objetos conceptuales como navegacionales, corresponde ser almacenada. También se debe decidir como será la apariencia de las pantallas que serán mostradas a los usuarios.

Si bien el ideal sería que los diseños desarrollados en cada una de las etapas puedan ser implementados sin problemas, resulta no ser así en las aplicaciones reales. Cuando se deciden las tecnologías y el desarrollo de la etapa de implementación comienza a ser desarrollada es indispensable, en general, tener que modificar o agregar objetos al diseño conceptual o modificar algún otro modelo para que la implementación pueda ser llevada a cabo. Esto implica modificar diseños ya cerrados para adecuarlos a las tecnologías. En general, es muy difícil que se logre la implementación completa de la aplicación sin tener que realizar ningún cambio que se deba hacer para ajustar el modelo a la implementación.

4.3 Utilización del Patrón MVC

El patrón MVC [39] fue utilizado originalmente en el lenguaje Smalltalk. Es utilizado en aplicaciones que contienen un modelo y una interfaz para observar los datos y lo que sucede en el modelo, e interactuar con el mismo. El principio básico de este patrón es la separación de responsabilidades dentro de las aplicaciones. La aplicación es dividida en tres componentes, el modelo, la vista y el controlador. Cada uno de estos componentes maneja un discreto conjunto de tareas que se deben realizar. La división de estas responsabilidades tiene como objetivo el de realizar la separación entre la vista y el modelo obteniendo de esta manera que el modelo sea más robusto, debido a que el desarrollador de la aplicación se encuentre menos expuesto a provocar cambios indebidos en el modelo, en caso de tener que modificar la vista. Además, con esta separación, se logra que estos cambios sean totalmente independientes unos de otros logrando reusabilidad, un mejor diseño y un mejor mantenimiento de la aplicación.

El modelo del MVC es el encargado de los datos y la lógica del dominio de la aplicación, este modelo no tiene ningún conocimiento sobre la interfaz de la aplicación, qué componentes ella contiene ni como es visualizada. La vista es la encargada de la parte gráfica de la aplicación, o sea, muestra los datos del modelo, contiene los componentes de interfaces como botones, listas, entre otros. Por último, el controlador, que es el que recibe los eventos de la interfaz realizados por el usuario, tales como la opresión de un botón, la selección de un elemento de la lista, entre otros. Su rol principal es el de avisarle al modelo que estos eventos ocurrieron, y el modelo en estos casos realizará las actividades necesarias en respuesta al evento producido.

La gran ventaja que propone el patrón MVC, es que se puede realizar una aplicación que posea un solo modelo que contenga diversas vistas sobre los mismos datos. Por ejemplo, un modelo de un termómetro donde una de las vistas muestre un visor con la temperatura en grados centígrados, otra vista en Celsius y otra muestre el gráfico de un

termómetro. En este caso, se tiene un modelo con diferentes vistas que muestran los datos de una misma aplicación. Por este motivo, se considera un patrón clásico para aplicaciones que contienen muchas vistas, las cuales se encuentran como dependientes del modelo. En las aplicaciones como éstas, el modelo posee la responsabilidad de notificar a sus dependientes, las vistas, cuando se realice algún cambio en el mismo. Realizando esta acción, el modelo avisa a sus vistas que deben actualizar su interfaz. Otra ventaja importante a destacar es que los cambios que son realizados en el modelo pueden ser realizados de forma independiente de la vista, y sin que ella se vea afectada en absoluto. Lo mismo ocurre de manera inversa si la vista es que la debe ser modificada.

Los flujos de comunicaciones de las interacciones de los tres componentes se describen a continuación [40]. Luego de esta descripción se observa la Figura 4.1 en donde estas comunicaciones se encuentran representadas.

1. La vista recibe datos de entrada del usuario y los comunica al controlador.
2. El controlador recibe, por parte de la vista, la notificación del evento producido por el usuario.
3. El controlador interactúa con el modelo en respuesta a al evento realizado por el usuario que le fue comunicado por la vista.
4. El modelo cambia basándose en las actualizaciones que son comunicadas por el controlador.
5. El modelo notifica a la vista o vistas que hubo cambios que se realizaron y que las afectan, debiendo ellas efectuar las actualizaciones correspondientes.
6. La vista actualiza la interfaz del usuario para que se observen los datos actualizados obteniéndolos del modelo. Interactúa con el modelo solo para pedirle datos.

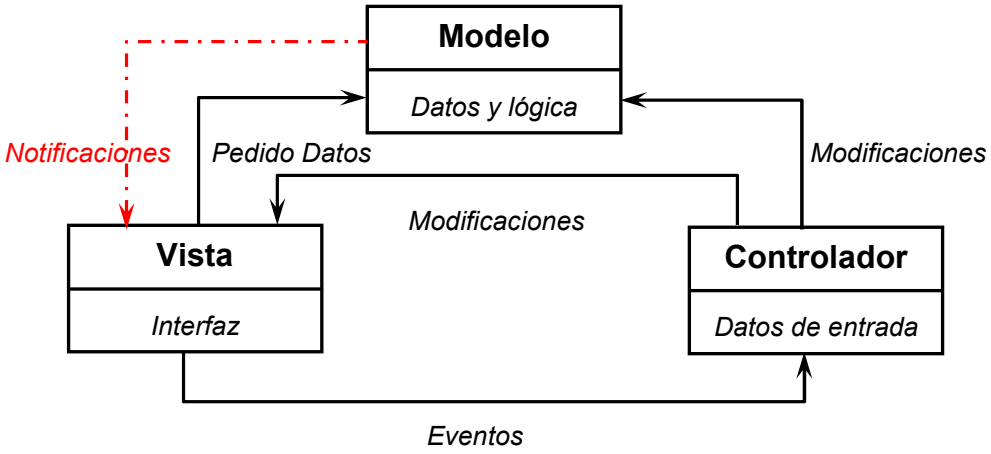

Figura 4.1: Modelo del patrón MVC.
(En la figura la línea rojo denota las notificaciones por dependencia).

Si bien se puede observar este modelo presentado en la Figura 4.1, en algunos casos se pueden observar implementaciones del MVC diferentes, en los cuales la vista tiene conocimiento directo con el modelo, u otra posibilidad es que la vista no tenga conocimiento del controlador. En estos casos, se pueden realizar algunas pequeñas variantes a este patrón, pero a pesar de dichas variantes la funcionalidad y la división de las responsabilidades con todas las ventajas se siguen manteniendo intactas.

Como conclusión, se deben destacar beneficios importantes que proporciona la utilización de este patrón en el desarrollo de las aplicaciones. La separación de responsabilidades que proponer logra una gran independencia del modelo con la vista, pudiéndose implementar y ser modificados en forma separada y por diferentes personas.

Provee además, la utilización de un mismo modelo para diversas vistas que obtienen los datos del mismo, solo que lo muestran de diferente manera o distintos datos. Otra ventaja que presenta es la reusabilidad. Por ejemplo, una vista desarrollada para un modelo puede ser reutilizada para otro. Y por último, un aspecto muy importante es que los desarrolladores pueden trabajar sobre uno solo de los componentes de la aplicación centrando toda su atención en el mismo sin tener que preocuparse por los demás componentes. Estas son las grandes ventajas que provee el uso de este patrón, lo cual lo ha llevado a ser uno de los más conocidos y utilizados en las aplicaciones que poseen un modelo y una interfaz que muestra los datos del mismo.

Este patrón surgió con las aplicaciones de escritorio, pero al aparecer las aplicaciones Web, fue aplicado también a éstas realizando pequeños cambios. Las aplicaciones Web complejas continúan siendo más difíciles de diseñar que las aplicaciones tradicionales de escritorio, el patrón MVC se presenta como una solución para ayudar a disminuir dicha complejidad. Se presentan más detalles sobre este tema en la sección 4.5.

4.4 Patrones de Personalización

En esta sección, se presentan y describen patrones de diseño definidos en [34] para aplicaciones personalizadas. Dichos patrones son encontrados en una gran mayoría de las aplicaciones personalizadas. Son presentados como uno de los conceptos de diseño que se utiliza para realizar las aplicaciones personalizadas, proporcionando una noción más concreta de la estructura y un conocimiento del comportamiento de la aplicación. Los patrones que se exponen se centran en qué es lo que se personaliza, sin describir cómo esto se realiza. Éstos son: Link Personalizado, Contenido Personalizado, Estructura Personalizada y Client-side Personalizado.

4.4.1. Link Personalizado

Es un patrón que adapta la topología de navegación de un sitio a las preferencias y necesidades de sus usuarios. Esto sucede cuando los links del usuario que accede a la aplicación pueden diferir de los link presentados a un usuario diferente. Por ejemplo, en las aplicaciones de comercio electrónico, cuando se realizan las recomendaciones de productos, se presentarán links a productos relevantes para el usuario basándose en los intereses que ha demostrado en recorridos anteriores por el sitio. Estos links, probablemente, diferirán entre los usuarios dependiendo de sus preferencias. O sea, los links de recomendación son presentados a todos los clientes del sitio, pero dichos link navegan a diferentes páginas dependiendo de las preferencias de los usuarios. En las siguientes figuras se muestra un ejemplo de este tipo de recomendación. En la Figura 4.3, se puede observar el link para navegar a las recomendaciones para un usuario particular. La Figura 4.3, muestra los productos obtenidos en base a preferencias del usuario.

Link Personalizado, puede implicar incluir en la aplicación algoritmos complejos para resolver las preferencias de los usuarios, como sucede en el caso del ejemplo anterior de las recomendaciones. En otros casos, los links personalizados pueden ser obtenidos ejecutando métodos simples en los objetos del modelo. De estos modos, se logran los resultados de los links que formarán la topología de navegación del sitio que recorrerá cada uno de los usuarios.

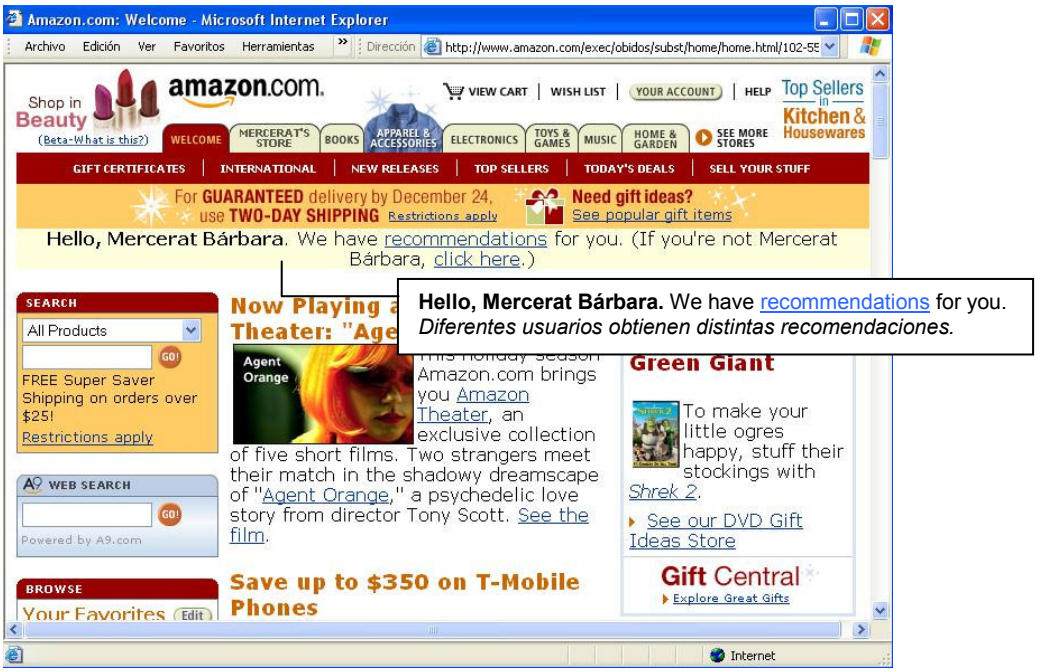

Figura 4.2: Link Personalizado en Amazon.

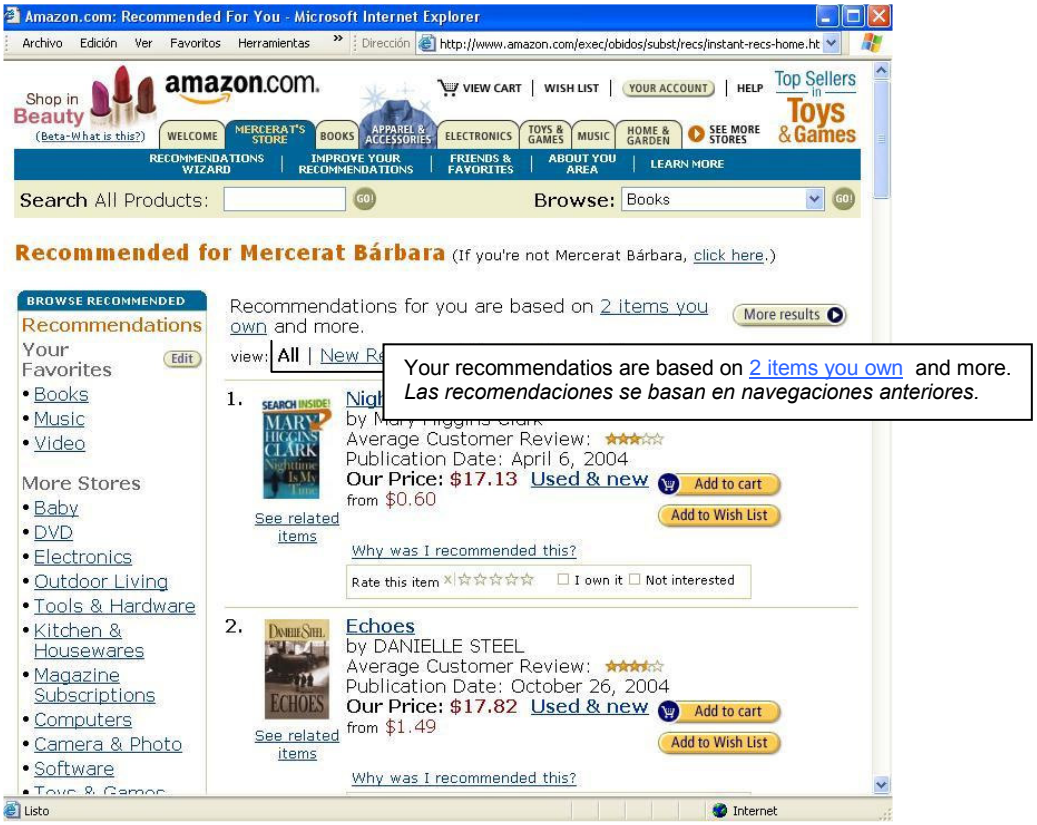

Figura 4.3: Página de recomendaciones de Amazon.

4.4.2. Contenido Personalizado

Su propósito es personalizar el contenido de las páginas. Se basa en mostrar diferente información a los distintos usuarios. Un ejemplo de este patrón se presenta en los sitios de comercio electrónico, en donde a los usuarios se les puede mostrar un precio diferente sobre el mismo producto, dependiendo del tipo de cliente que éste sea. Si es un cliente que realiza compras frecuentemente, en algunos sitios es muy posible que se le aplique un descuento al total de la compra, lo que no ocurrirá en el caso que el usuario compre esporádicamente. Otro ejemplo concreto se presenta en un sitio de venta de componentes de computadoras. El usuario que se loguea al sitio puede ser un cliente particular que compra un producto, o un comercio que vende al público componentes de computadoras y compra en cantidad. En estos casos, los precios presentados a cada uno de los usuarios es diferente, el cliente solo compra unos pocos productos y el otro compra en cantidad, por lo cual al cliente que es del comercio se le aplicará un precio por compra mayorista, este descuento no es aplicado a los clientes particulares. Se observa claramente que se personaliza el contenido de los atributos a mostrar, en este ejemplo, el precio. Otro ejemplo de este patrón, se encuentra en las intranets, donde distintos usuarios de la intranet acceden a información diferente cuando acceden a un mismo ítem. Por ejemplo, en una intranet de una compañía de teléfonos móviles, cuando el empleado de atención de llamadas de reparación busque información sobre reparación del aparato, se le mostrará la dirección del centro de reparación a dónde debe enviarlo. En cambio, cuando un empleado del área de reparación busque información para el mismo procedimiento, la página será diferente, se le indicará qué es lo que debe reparar y cuál es el problema del aparato. En conclusión, con este patrón la topología de navegación no se encuentra afectada, solo el contenido de la información.

4.4.3. Estructura Personalizada

Su objetivo es el de delimitar el espacio navegacional a aspectos en los que el usuario se encuentra interesado. Ciertas aplicaciones Web, contienen enormes cantidades de elementos, entre ellos se encuentran mucha información y gran variedad de servicios, lo cual provoca que el usuario se pierda dentro del sitio y por este motivo, posiblemente, no obtenga la información buscada. Esto se produce porque no tiene una visión clara del sitio, dado que se presenta mucha información, y dentro de ella mucha de la cuál no es de interés para el mismo. El problema es, cómo lograr una buena estructura de la información del portal para brindarle al usuario libertad de navegación sin provocar que se pierda con la cantidad de información exhibida. Para solucionar este conflicto de excesiva información surgió el patrón Estructura Personalizada, que personaliza la estructura del sitio, o deja que el usuario lo haga. Es decir, se provee acceso a módulos de información, cada uno de los cuales puede estar compuesto por otros módulos. Se seleccionan y exhiben solo los módulos que son de interés para el usuario, y de estos módulos solo se muestra la información que el usuario desea examinar. Un ejemplo de este patrón se encuentre en el sitio MyYahoo, como se muestra en la Figura 4.4 y 4.5. La Figura 4.4, muestra las posibilidades de información dentro del sitio y como éstas pueden seleccionarse. En la Figura 4.5, se muestra el resultado de una pequeña selección de datos a mostrar. De este modo, se simplifica la apariencia de las páginas solo presentándole la información de su interés. Este tipo de personalización puede ser realizada por el usuario, en la cual el mismo selecciona los módulos que desea ver, como lo visto en el ejemplo, o de manera que el sitio es el encargado de mostrar o no los módulos del usuario dependiendo de las preferencias del mismo.

Figura 4.4: Selección de estructura Personalizada en MyYahoo.

Figura 4.5: Estructura Personalizada en MyYahoo.

4.4.4. Client-side Personalizado

Este patrón se basa en que una aplicación Web brinde diferente información personalizada dependiendo del sitio desde el cual se accede a la misma. Este patrón es utilizado en aplicaciones que utilizan servicios de otras aplicaciones. Por ejemplo, en ciertos sitios se utiliza un servicio de otro sitio Web para proporcionar búsquedas más eficientes, o brindando algún servicio que el sitio por el cual se está navegando no lo provee. Este ejemplo se puede observar en www.CNN.com que utiliza la búsqueda del sitio Yahoo. En este patrón se personaliza de una manera diferente a la descripta en los patrones anteriores. En este caso, se personaliza el sitio que provee el servicio adaptándolo al sitio que lo brinda. No es una personalización que se relaciona a un usuario particular, sino una personalización que se adapta a los diferentes contextos desde los cuales se accede a dicha aplicación.

Existen dos modalidades de uso de Client-side Personalizado. Una, es que la aplicación cliente muestre una ventana de información provista del servicio, como en el caso del ejemplo de la búsqueda en www.CNN.com, que se muestra en la Figura 4.6. La otra modalidad, es personalizar lo que visualiza la aplicación que provee el servicio cuando se navega hacia ésta desde otra aplicación (aplicación cliente). Es decir, ésta muestra diferente información dependiendo del sitio al que le esté proporcionando el servicio. Un caso de este tipo de personalización, es el que se muestra a continuación en las Figuras 4.7 y 4.8. En este ejemplo, no es necesario navegar hacia el otro sitio sino que se tiene una pequeña visión del mismo proporcionando el servicio solicitado.

Figura 4.6: Yahoo como servicio de búsqueda CNN.

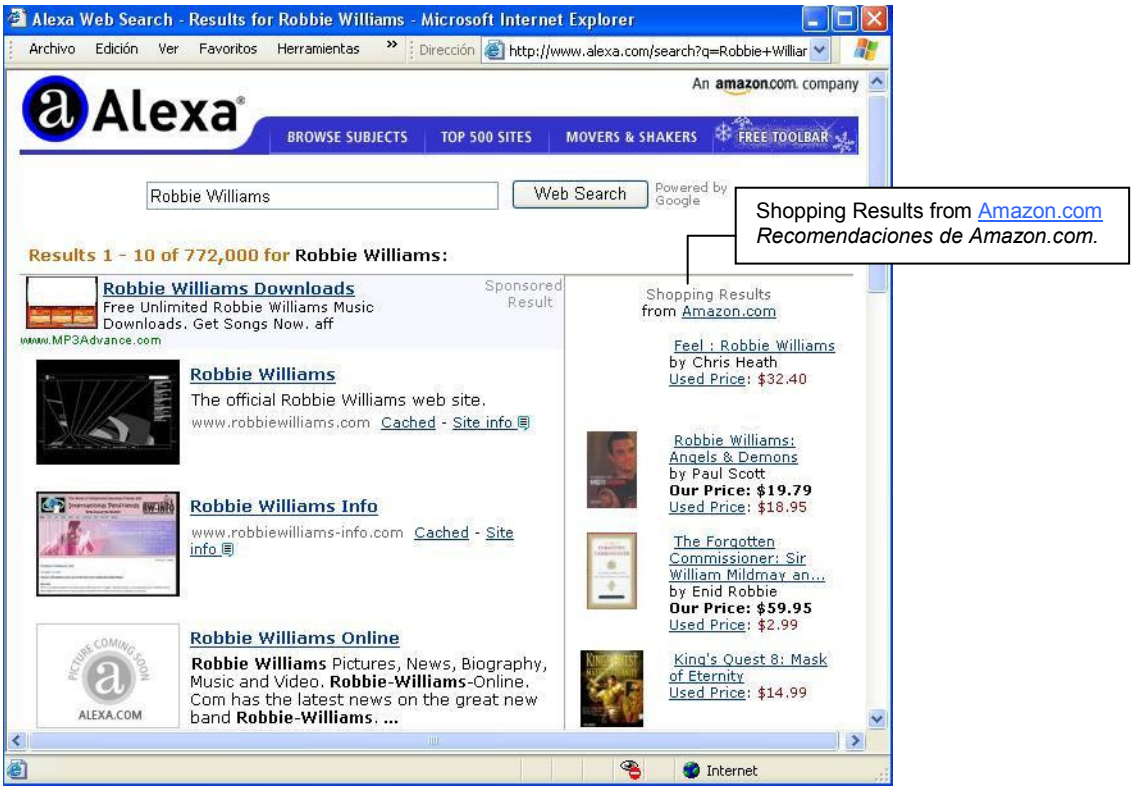

Figura 4.7: Búsqueda en Amazon, brindando el servicio.

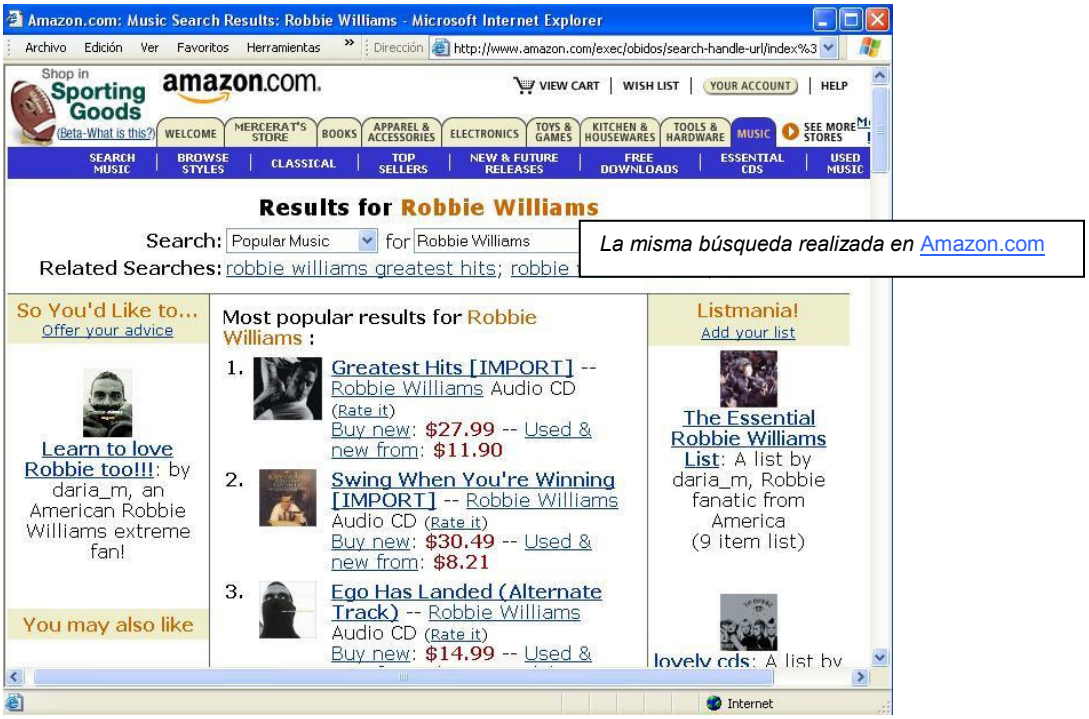

Figura 4.8: Búsqueda dentro de Amazon.

4.5 Aportes de la Metodología y Patrones a la creación de la Arquitectura

Esta sección pretende explicar y presentar cada una de las características o aportes que realizaron la metodología OOHDM, el patrón MVC y los patrones de personalización Web, para el desarrollo de la arquitectura propuesta.

Como se observa en la sección 4.2, OOHDM es una metodología que no contempla aspectos de personalización en el desarrollo del diseño de las aplicaciones hipermediales. Sin embargo, aportó ideas sobre la separación de responsabilidades en la aplicación demostrando que la separación del modelo conceptual, de la interfaz y del modelo navegacional proveen un mejor entendimiento y flexibilidad de las aplicaciones, ya que esta separación se realiza en base a la división de responsabilidades. En la arquitectura propuesta se puede observar claramente cual fue su influencia, dado que el modelo de la aplicación de la arquitectura realiza su división en base a esta metodología, adicionándole a dicho modelo otras responsabilidades más para lograr los aspectos que contemplan la personalización.

Con respecto al MVC, sucede que la implementación tradicional del mismo no se realiza de la misma manera en las aplicaciones Web. Se produjeron dos cambios principales en su implementación. Uno, con respecto a la comunicación entre las componentes del mismo; y el otro cambio, con respecto a la tecnología utilizada para implementarlo [41]. El cambio que afecta a la comunicación ocurre por el hecho de que en otro tipo de aplicaciones el modelo tiene como dependientes a sus vistas. En contraste, en las aplicaciones Web el modelo no tiene ningún control sobre sus vistas, no las conoce, para él solo son requerimiento que le piden y él responde. Por lo cual, en estas aplicaciones cuando se implementa este patrón se elimina del componente modelo la responsabilidad de avisar a sus vistas cuando deben ser actualizadas [42]. En la Figura 4.1 la flecha punteada de color roja no existiría. La comunicación de estas aplicaciones se basa en que la vista realice pedidos o requerimientos y la aplicación responda a dichos pedidos. La arquitectura propuesta utiliza el patrón MVC para establecer la comunicación con los usuarios, proveyendo cada uno de los componentes que se describen en este patrón, una vista, un controlador y un modelo, cada uno con las responsabilidades que deben cumplir. Otro notable cambio que se produjo fue con respecto a la implementación de la vista, que la misma pasa a utilizar diferente tecnología en la implementación que el modelo y el controlador. Este cambio se produce dado que las aplicaciones Web utilizan como vistas para los usuarios páginas HTML, WML, entre otros, para lograr que los usuarios vean las páginas a través de un browser, mientras que los componentes modelo y controlador se implementan con leguajes de objetos, entre otros, para lograr la realización de la lógica de la aplicación. Este cambio se efectuó cuando las aplicaciones Web fueron tornándose dinámicas y más complejas. En el área de desarrollo estos cambios denominaron un nuevo modelo de MVC, llamado MVC Modelo 2 [41], pero existen discusiones sobre esta denominación nueva dado que algunos nombran MVC Modelo 2 a la implementación del MVC utilizando páginas JSP, y otros realizan otras diferencias, por lo cual en este trabajo se seguirá llamando patrón MVC dado que el mismo se encuentra orientado a presentar una arquitectura y no una implementación específica, se presentará una implementación como posible pero no como definitiva.

Finalmente, los patrones de personalización Web brindaron una buena base para poder contemplar en el desarrollo de la arquitectura que la misma sea capaz de soportar dichos patrones tan utilizados en las aplicaciones personalizadas, asimismo, se tuvieron en cuenta otros aspectos a ser personalizados concernientes a estas aplicaciones. Una vez terminada la arquitectura se utilizaron para poder fundamentar que es capaz de soportar los aspectos de personalización de una manera sencilla, demostrando su potencialidad.

Si bien anteriormente se presentaron los aportes que realizaron a la arquitectura

propuesta, en los capítulos siguientes cuando se presenta la arquitectura y sus detalles, se irán destacando y entendiendo mejor como fue que esta metodología y los patrones fueron tan importantes e influyentes para el desarrollo de la misma.

Capítulo 5

Diseño General de la Arquitectura

Durante el desarrollo de los capítulos anteriores se fueron presentando las particularidades y aspectos pertenecientes a las aplicaciones personalizadas. En este capítulo, se presentan las primeras características y un primer nivel abstracción de la arquitectura modular propuesta para resolver el desarrollo de las aplicaciones Web personalizadas.

Como un primer paso a la presentación de la arquitectura, en la sección 5.1, se realiza una exposición de cómo va a ser presentada la arquitectura y en la siguiente sección, las características generales que presenta. Posteriormente se presenta el desarrollo del primer nivel de la arquitectura. Este nivel efectúa la utilización del patrón MVC, y se describe como es la ubicación que ocupan los componentes del MVC entre el cliente y el servidor, y además, se exponen las razones por las cuales se realizó la ubicación de ese modo, en la sección 5.3. Se introdujeron en esta sección dos subsecciones destinadas a la explicación de la utilización del MVC dentro de la arquitectura. Una de ellas, la subsección 5.3.1 detalla las responsabilidades de cada uno de los componentes y se describen qué funcionalidades poseen cada uno de ellos, cómo es realizada la interacción de los usuarios con la aplicación y cómo se realiza la comunicación entre componentes para resolver los pedidos realizados por los usuarios. Mientras se realiza la exposición de estos temas se exponen las ventajas que brinda la utilización del patrón MVC. En la sección siguiente, subsección 5.3.2, se señala una característica que poseen las interfaces de hoy en días y cómo es resuelta por la arquitectura. En este capítulo se dedica la sección 5.4 a presentar una implementación posible para demostrar que este primer nivel de la arquitectura puede ser llevado a cabo con tecnologías utilizadas hoy en día para el desarrollo de aplicaciones Web. Se menciona como posible, dado que se seleccionó una tecnología para hacerlo y puede ser implementada con otras existentes. Finalmente y como cierre de este capítulo, en la sección 5.5 se exponen las conclusiones obtenidas de las descripciones, características y ventajas de este primer nivel.

5.1 ¿Cómo será presentada la Arquitectura?

Entre éste y el siguiente capítulo se presentará la arquitectura propuesta, pudiendo observar como la metodología y los patrones expuestos en el capítulo 4 influyeron en el diseño y desarrollo de la misma. Esta arquitectura es presentada en dos niveles de abstracción. El primer nivel que será presentado en este capítulo es el nivel más externo de la arquitectura en el cual se presentará como se realiza la interacción de los usuarios con la aplicación. Este nivel efectúa la utilización del patrón MVC indicando qué funcionalidad poseen cada uno de sus componentes, cómo se desempeña la comunicación entre ellos y la ubicación que ocupan dichos componentes entre el cliente y el servidor. Posteriormente, el segundo nivel presentado en el capítulo 6, es el nivel en donde se realiza la descripción completa del componente modelo del MVC que en la arquitectura propuesta posee un papel fundamental. Este modelo se encuentra compuesto por módulos internos, cada uno con su comportamiento y responsabilidades diferentes que se combinan para lograr funcionar como modelo de la aplicación personalizada. En dicho capítulo se realiza una descripción detallada y completa de sus funcionalidades e interacciones que realiza para lograr cumplir con sus responsabilidades.

Cómo se observará en los siguientes capítulos, la arquitectura expuesta se encontrará acompañada de secciones de implementación, habrá una sección que implemente el primer nivel de abstracción presentado, y luego otra que exponga una implementación del segundo nivel. Cada una de estas explicaciones se realizará posteriormente a haber finalizado con la exposición de las funcionalidades y responsabilidades de cada una de las partes que componen la arquitectura. Esta exposición de implementación es debida a que se tomó la decisión de realizar una posible implementación, no definitiva, utilizando tecnología actual para demostrar que lo que se propone en este trabajo no permanece pensado solo como una idea teórica, sino que las aplicaciones Web personalizadas pueden ser llevadas a la práctica basando su realización en la arquitectura propuesta utilizando tecnologías muy empeladas en estos días. La implementación que se expone en este trabajo es una de las posibilidades existentes entre otras, que no serán expuestas por el motivo de que no se quiere desviar el objetivo de este trabajo.

A continuación se comienza con la presentación de la arquitectura, presentando como primera instancia algunos aspectos generales de la misma para luego ir incrementando el grado de detalle.

5.2 Características generales

Para el desarrollo de la arquitectura, el trabajo de investigación se basó en las ideas propuestas por OOHDM y el patrón MVC de separar las tareas en base a las responsabilidades que se deben cumplir en la realización de aplicaciones Web, en este trabajo específicamente aplicaciones Web personalizadas, logrando con dicha separación independencia y facilidad de desarrollo y mantenimiento. Un punto importante que debe ser señalado es que para alcanzar la realización de la misma se tuvieron muy en cuenta los conocimientos de las posibilidades de dinamismo y facilidades de desarrollo que brindan las tecnologías que surgieron en los últimos años para el desempeño de aplicaciones Web. Otro de los puntos a destacar, es que la personalización de la aplicación requiere de datos de los usuarios y ellos de un almacenamiento y procesos de manipulación de los mismos dentro de la aplicación, por este motivo se partió de la base que se deben tener perfiles de usuarios para dicho requisito fundamental de este tipo de aplicaciones.

Con el desarrollo de esta arquitectura se desea contemplar todos los aspectos personalizables que fueron descriptos en capítulos anteriores, consiguiendo de este modo, abarcar todas las posibilidades que se presentan en las aplicaciones Web personalizadas. Si bien se representa orientada a una aplicación de comercio electrónico

se explicará como puede ser empleada y adaptada para otras aplicaciones con diferentes objetivos.

Como se observará, en el desarrollo de estos últimos capítulos que describen la arquitectura propuesta, ésta se encuentra compuesta por diferentes módulos, los cuales cada uno posee una funcionalidad particular e indispensable. El principal éxito de la arquitectura es la separación de responsabilidades que poseen los módulos que la componen, logrando con dicha modularidad el objetivo de que dichos módulos puedan ser desarrollados y diseñados de manera separada, además logrando que los cambios realizados luego de creada la aplicación puedan ser realizados, y poder desempeñarlos de manera sencilla y fácil. Esta separación no implica que los módulos no dependan unos de otros, pero se puede lograr diseñar uno de los módulos, desarrollarlo y luego diseñar los otros en base al comportamiento del que ya fue creado con anterioridad, logrando de este modo que las tareas de diseño, implementación y mantenimiento de la aplicación sean realizadas de una forma más sencilla, y solo se centre la atención en cada uno de los módulos a desarrollar.

Por último, cabe destacar que el tema de persistencia de los datos no será tenido en cuenta en el desarrollo de esta arquitectura, dado que este tema es tratado e investigado en otros trabajos de estudio. Solo se hará referencia al mismo como una base de datos que contiene los datos que serán almacenados, pudiendo ser implementado con otro tipo de almacenamiento existente, y el modelo de la aplicación interactuará con el mismo para persistir y obtener los datos que sean necesarios persistir, como se verá en los gráficos de la arquitectura.

5.3 Utilizando el Patrón MVC en la Arquitectura

Como fue explicado en la sección 4.5 el MVC clásico fue adaptado para ser utilizado en las aplicaciones Web, conservando sus tres componentes, Modelo, Vista y Controlador, pero fueron efectuados cambios en cuanto a la manera de realizar la comunicación entre ellos y en cuanto a la implementación. Respecto al cambio en la comunicación, la implementación del Modelo no posee la responsabilidad de notificar a las vistas de los cambios que se produjeron en el mismo. Este cambio se debe al tipo de actividad que realizan las aplicaciones Web dado que es imposible tener el control sobre todas las vistas que los usuarios se encuentran accediendo con respecto a una única aplicación. En cuanto al cambio de implementación, la tecnología empleada para implementar el componente Vista pasó a ser diferente con respecto a la tecnología utilizada para el Controlador y el Modelo. La visualización de las aplicaciones Web es realizada por los usuarios a través de un browser que muestra el contenido producido por las aplicaciones en respuesta a un pedido. Esta visualización que se realiza en los browsers se efectúa a través de páginas de internet, que son páginas HTML, las cuales no poseen ningún tipo de procesamiento, son solo visuales conteniendo links que proveen el acceso a otras páginas. En cambio, es necesario que el Controlador y el Modelo contengan lógica de la aplicación, puedan manipular y procesar la información y generar a su efecto respuestas a los usuarios. Para lograr estas responsabilidades, estos componentes deben poseer programación que se encargue de todas las tareas de procesamiento, por lo cual HTML no alcanza para realizar estas tareas y se debe emplear otra tecnología.

En las primeras aplicaciones Web creadas en el espacio de Internet bastaba solo con las páginas HTML que no contenían ningún tipo de procesamiento por lo cual eran totalmente estáticas. En estas aplicaciones no era utilizado el patrón MVC ya que toda la lógica de la aplicación era encontrada en las mismas páginas HTML y no existía la idea de separación de responsabilidades. Con el progreso de las aplicaciones Web, comenzaron a surgir conceptos tales como separación de responsabilidades y reusabilidad en el diseño e implementación de las aplicaciones Web y el progreso de

éstas en cuanto a la realización de crear aplicaciones personalizadas. Con estos progresos no alcanzaba con la realización de páginas HTML únicamente, sino que se necesita realizar programación más compleja y mejores diseños para lograr realizar aplicaciones más complejas y dinámicas [43]. De la mano de estos conceptos surgen las diferentes tecnologías hoy en día utilizadas en las aplicaciones Web, como son JSP, Servlets, XML, XSL, Java, el MVC adaptado a las aplicaciones Web, patrones Web que surgieron, ASP, PHP, EJB, .Net, y una variedad más de nombres de tecnologías y lenguajes a ser nombrados que poseen características para crear estas aplicaciones [44].

Estando al corriente de los avances logrados por las metodologías, patrones, y tecnologías se logró realizar una arquitectura como la expuesta en este trabajo que incluye metodologías, patrones, y tecnologías muy utilizadas en estos días que brindan varias ventajas. Con el principal objetivo de lograr independencia entre las responsabilidades de la aplicación y teniendo en cuenta las ventajas ya señaladas que esta separación provee, se utiliza en la arquitectura propuesta el patrón MVC, como puede observarse en la Figura 5.1.

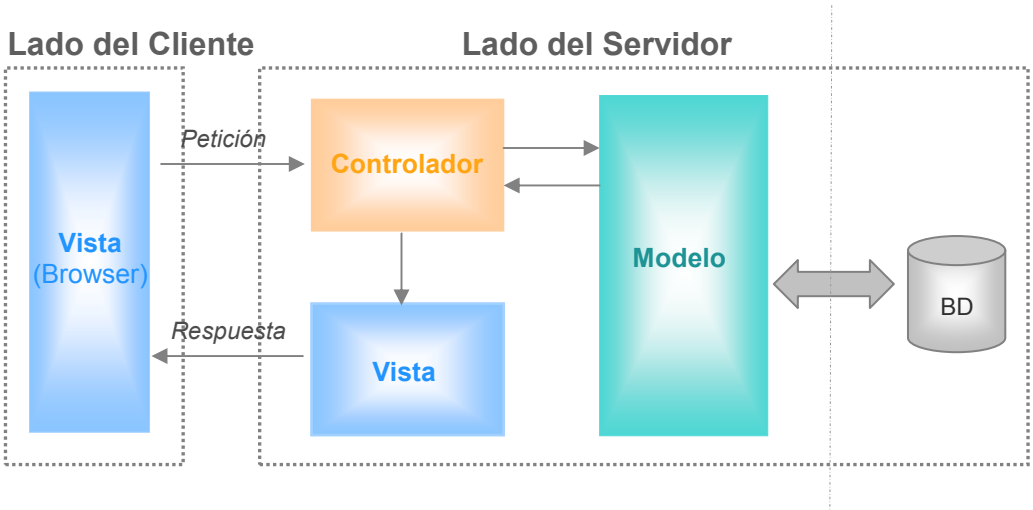

Figura 5.1: Patrón MVC en la arquitectura.

El MVC es el encargado de realizar la comunicación entre los usuarios y la aplicación. Los roles y responsabilidades de cada uno de los componentes no cambian en su empleo en esta arquitectura, siguen manteniéndose iguales que en el MVC clásico, solo se han realizado los cambios anteriormente descriptos para adaptarlo a una aplicación Web.

Los pedidos efectuados por los usuarios son recibidos por el MVC que cumple con la responsabilidad de resolverlos proporcionando como resultado una respuesta de salida. El pedido realizado por los usuarios atraviesa los tres componentes que componen el MVC, cumpliendo cada uno de ellos con su responsabilidad, realizando procesos sobre el pedido y generando como resultado una respuesta a la solicitud realizada.

Como se muestra en la Figura 5.1, el Controlador, el Modelo y una parte de la Vista que componen el MVC junto con el almacenamiento de los datos, se encuentran ubicados del lado del servidor, del lado del cliente solo se tiene una parte de la Vista que es la visualización de la aplicación, la cual no posee procesamiento, es el browser por el cual el usuario accede a la aplicación. Al ubicar los tres componentes en el servidor la aplicación evita colocar elementos en el cliente, de esta manera, si el usuario no permite la instalación de elementos en su máquina, no se corre el riesgo de que la aplicación no funcione. Además, se tiene la ventaja de que no se emplea espacio de la máquina del cliente, y el tema de privacidad de los datos es manejado por los realizadores de la aplicación, y no se deposita esta responsabilidad en el usuario. Otra ventaja a destacar

con esta decisión es con respecto al flujo de información, una vez realizado el pedido del usuario la información circula a través de componentes en el servidor lo cual logra una ventaja en el tiempo de obtención de datos ya que la aplicación deja de interactuar con el usuario. Si se produjera esa situación el tiempo de latencia sería mayor, por lo cual el tiempo de respuesta a la solicitud realizada también aumentaría, y la perfomance de la aplicación se vería afectada notablemente.

La vista que se observa del lado del cliente, es solo la visualización de la aplicación, las páginas que serán presentadas a los usuarios. Esta misma no implica procesamiento del lado del cliente, por lo cual, no se le impone al usuario capacidad de procesamiento ni de almacenamiento disponible en su máquina.

En la siguiente sección se expone qué responsabilidades poseen cada uno de los componentes y cómo es realizada la comunicación entre los usuarios, la Vista, el Controlador y el Modelo.

5.3.1. Flujo de la Información

A continuación se describe cómo se realiza la comunicación entre los usuarios y la aplicación, además, se describen qué tareas realizan cada uno de los componentes del patrón. Se pueden observar en esta sección los cambios realizados al MVC clásico en cuanto al aspecto de la comunicación entre los componentes y, a pesar de no hablar de tecnologías de implementación, se describe como las diferentes tecnologías que se emplearán en la implementación del patrón afectan a este diseño del MVC, presentándose como otro de los cambios que se producen en el MVC para ser utilizado en aplicaciones Web.

La comunicación es iniciada por los usuarios realizando pedidos a la aplicación. La misma resolviendo estos pedidos les retorna respuestas que son visualizadas por los usuarios a través del browser. En la Figura 5.2 se pueden observar flechas con números que indican el inicio de la comunicación desde los usuarios a través del browser, y el paso de la información por cada uno de los componentes para luego finalizar donde se inició, en el browser.

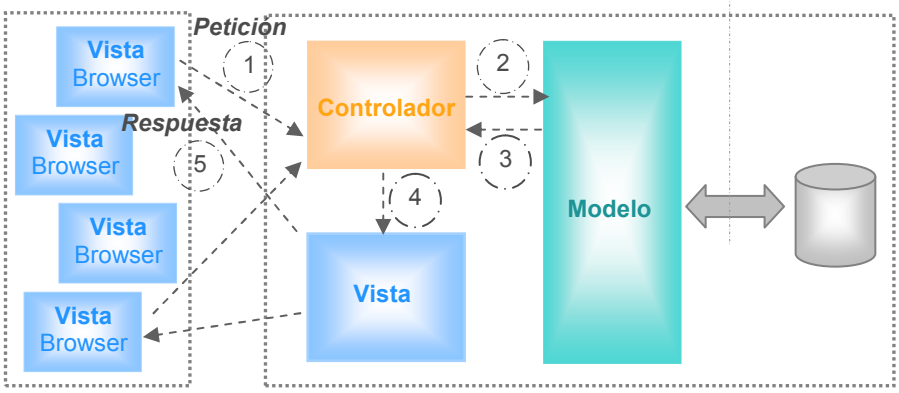

Figura 5.2: Comunicación entre los componentes del MVC.

Como primera descripción del MVC se debe destacar que el componente Vista se encuentra compuesto por dos partes. Una de ellas, se encuentra del lado del servidor y la otra parte del lado del cliente, como se describió en la sección anterior y como se puede observar en la Figura 5.2. Las dos partes poseen funcionalidades totalmente diferentes y sus tecnologías serán diferentes también, dado que cada una, si bien forman parte del componente Vista en el MVC, tienen un rol diferente. Se puede marcar una gran diferencia entre ellas, que es que la Vista que se encuentra del lado del servidor procesa datos para obtener un resultado que será mostrado por la Vista del lado del cliente, en cambio, la Vista del cliente no realiza ningún tipo de procesamiento, solo visualiza. Por

este motivo evidente, una de las partes utiliza tecnología diferente con respecto a la otra. Otra diferencia a destacar, es que la Vista que se encuentra del lado del servidor es una sola, en cambio puede haber varias Vistas que se encuentran del lado del cliente. Se comprenderán más claramente estas diferencias y particularidades de cada parte en la siguiente descripción de la comunicación y responsabilidades de los componentes del patrón MVC.

El usuario accede a la aplicación a través de un browser que forma parte de la Vista del MVC. Esta vista posee la responsabilidad de la visualización de la aplicación, la información personalizada en respuesta al pedido realizado, exhibe las páginas que serán mostradas a cada uno de los usuarios, a través de las cuales los usuarios interactúan con la aplicación y realizan los pedidos *(Petición)*. Estas peticiones son producidas al realizar un click en un link de la vista, o el envío de un formulario que fue completado. Cada una de estas peticiones son derivada en una solicitud efectuada al Controlador *(1)*.

El Controlador recibe el pedido realizado por el usuario y debe enviarlo al Modelo *(2)*. Para lograr esta tarea, el Controlador antes de enviar el pedido debe interpretarlo y luego traducirlo en un mensaje que el Modelo de la aplicación pueda entender. Como se explicó en secciones anteriores, la Vista del lado del cliente *(browser)* en las aplicaciones Web utiliza una tecnología diferente al Controlador y al Modelo. Por este motivo, estas tareas de interpretación y traducción deben ser realizadas, dado que el pedido es efectuado generalmente como un requerimiento http que no es entendible directamente por el Modelo, entonces el Controlador deberá traducirlo para poder enviarlo al Modelo que se encuentra implementado en otra tecnología y no lo entendería si no se lo envían en un modo que pueda ser entendible por él. Asimismo, enviará dicho pedido al elemento del Modelo que le correspondiente la tarea de resolver el pedido realizado.

El Controlador por otra parte, tiene la responsabilidad de mantener y manejar las sesiones de los usuarios que acceden a la aplicación. Existen diferentes tipos de sesiones que puede llegar a mantener, de página, que se mantiene mientras el usuario navega por la misma página, de pedido, que durará desde que se realiza el pedido del usuario hasta que se le envía la respuesta, de sesión, que perdurará durante la sesión completa del usuario, y se terminará una vez que el mismo deje la aplicación, y por último de aplicación, que en raros casos es utilizada, que permite que la sesión permanezca abierta durante todo el tiempo de vida de la aplicación. Dependiendo de las características de la aplicación se tomará la decisión de qué sesión de usuario se utilizará y será el Controlador el encargado de cumplir con esta tarea.

El pedido que el Controlador realiza al Modelo, no solo indica la acción producida por el usuario enviándola al elemento del Modelo correspondiente, sino que además, envía parámetros con datos. Estos parámetros pueden incluir información tanto de la aplicación como particular de los usuarios. Los parámetros particulares de los usuarios son fundamentales para realizar la personalización de los mismos, ya que poseen información sobre la trayectoria que realizó el usuario, tiempo de permanencia en la página, preferencias sobre el color de fondo, fuente de los textos, entre otros. Estos datos serán procesados luego en el Modelo para posterior almacenamiento en los perfiles de usuarios, y de esta manera ir construyendo y manteniendo las preferencias y gustos de los mismos.

El componente más importante en la arquitectura propuesta es el Modelo, que en el capítulo siguiente será presentado con mayor detalle. Este componente es el responsable de resolver toda la lógica de la aplicación. Posee el dominio de la aplicación, así como la lógica de la misma. Es el encargado de resolver los pedidos realizados por los usuarios. Para realizar esta tarea, el mismo contiene encapsulado todo el comportamiento relacionado con el aspecto de personalización. Realiza sus procesamientos para resolver el pedido efectuado interactuando con módulos correspondientes a la personalización de los usuarios y elabora la respuesta basándose en ellos, y de este modo, obtiene respuestas

personalizadas para cada uno de los usuarios. Este componente encapsula la lógica correspondiente a la personalización, mientras que los componentes Vista y Controlador, no contienen ningún aspecto que haga referencia a que la aplicación es personalizada, o sea, sucede que estos dos componentes no se verán afectados por los aspectos de personalización. Utilizando las características y ventajas que son provistas por la separación del MVC toda la personalización a realizar se encuentra encapsulada en el Modelo, por lo tanto este componente es el más importante y más complejo.

El Modelo contiene los elementos del dominio y lógica de la aplicación, y en las aplicaciones de comercio electrónico es el que contiene las reglas de negocio. Es el encargado de resolver los pedidos de los usuarios, utilizando la lógica que posee, los elementos del dominio, las reglas de negocio y los aspectos de personalización. Para lograr esta tarea además, interactuará con la base de datos para obtener y almacenar los datos que fueran necesarios, y de este modo obtener la información para seguir procesando el pedido realizado por el usuario. En la Figura 5.2 se observa que el modelo interactúa con la base de datos, o con cualquier otro tipo de almacenamiento de datos. Una vez que el Modelo produce la respuesta personalizada, envía los datos correspondientes al Controlador en respuesta al pedido realizado por el mismo *(3)*.

Una vez que el Modelo le retorna al Controlador la respuesta al pedido realizado, este último le envía el resultado a la Vista que se encuentra del lado del servidor para su procesamiento *(4)*. Esta interacción intermedia entre el Modelo, el Controlador y luego la Vista, se realiza con el objetivo de que el Modelo se encuentre totalmente abstraído de la parte visual de la aplicación.

La Vista que se encuentra en el servidor, recibe del Controlador toda la información para conseguir armar la página personalizada de respuesta que será presentada al usuario que realizó el pedido. Ni el Modelo, ni el Controlador, aplican algún formato sobre la información de respuesta, por lo cual, cuando llega a la Vista no posee ningún formato determinado. Esto es debido a que como se explicó anteriormente, ni el Modelo ni el Controlador poseen la misma tecnología que la Vista que se encuentra del lado del cliente, y no es responsabilidad de ellos conocerla. Por lo cual, se delega esta responsabilidad en la Vista que se encuentra en el servidor siendo ésta la encargada de transformar la información de respuesta enviada por el Modelo a datos entendibles por los usuarios. De esta manera, se logra una total independencia entre el Modelo de la aplicación y la Vista que observan los usuarios. La información que la Vista transforma es enviada como respuesta a la Vista del lado del cliente *(5)*, que es visualizada a través de un browser. Luego el usuario realiza otra petición *(1)* y el circuito vuelve a comenzar hasta lograr una respuesta *(5)*.

Puede observarse que en esta sección queda un poco "mágico" el desarrollo de cómo el componente Modelo resuelve los pedidos realizados por los usuarios. El motivo de que esto suceda se basa en el hecho de que el siguiente capítulo se encuentra dedicado completamente a su descripción, dado que es el componente más complejo y principal que fue desarrollado en este trabajo.

5.3.2. Diferentes tecnologías utilizadas en la interfaz

Las aplicaciones Web han sido históricamente representadas a través de páginas HTML. Si bien el lenguaje HTML continúa siendo la interfaz dominante para las aplicaciones Web, con el auge, el avance de las tecnologías y la complejidad que adquirieron estas aplicaciones, surge la posibilidad de realizar la visualización de dichas aplicaciones a través de diferentes tipos de interfaz [45]. De la mano de esta nueva peculiaridad, nace la necesidad de que las aplicaciones de hoy en día deban poseer la capacidad de soportar múltiples tipos de usuarios con múltiples tipos de interfaces, ver Figura 5.3. Se pueden mencionar a modo de ejemplo diferentes tipos de interfaz, Macromedia Flash, y alternativas como lenguajes markup tales como XHTML, XML/XSL, WML y Web services .

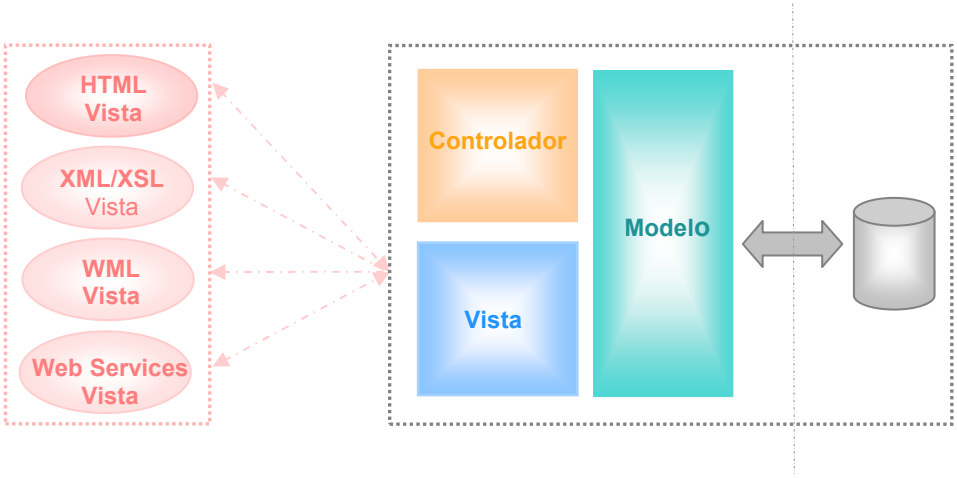

Figura 5.3: Existen diferentes tipos de Interfaz.

El problema a resolver es lograr visualizar los mismos datos utilizando diferentes interfaces. La arquitectura propuesta lo resuelve sin realizar ningún cambio en el diseño de la misma. Esto se debe a que uno de los principales beneficios que provee el patrón MVC es el de tener diferentes Vistas para un mismo Modelo, y es precisamente lo que ocurre en este caso. Por lo tanto, para resolver dicha cuestión se debe adicionar funcionalidad en el Controlador y en la Vista que se encuentra del lado del servidor. El Controlador debe tener la capacidad de interpretar el pedido realizado por el usuario conteniendo la funcionalidad de interpretarlo, pudiendo venir el pedido en un formato que no sea únicamente HTML. De este modo, una vez interpretado se enviará el pedido al Modelo como explicó en la sección anterior. Cuando la respuesta llegue al componente Vista, se realizará la transformación de la información que le fue enviada por el Controlador a la misma tecnología con la cual fue realizado el pedido. La Vista deberá contener la funcionalidad para poder transformar la información al formato requerido por el usuario.

Al poseer un Modelo abstraído totalmente de la tecnología de interfaz que es utilizada por los usuarios, no se le debe realizar ningún tipo de cambio, ya que en él se encuentra solamente la funcionalidad propia de la aplicación sin tener en cuenta aspectos de interfaz. Su responsabilidad es la de resolver el pedido realizado utilizando el dominio de la aplicación, y brindándole a la interfaz toda la información que necesite para armar el resultado. Delegando la responsabilidad de la transformación de dicha información a la Vista, que sabrá como procesarlos y enviarlos a los usuarios en un formato entendible por los mismos.

5.4 Implementación del MVC

Esta sección presenta una implementación del primer nivel de la arquitectura, presentando en las secciones anteriores. Si bien se presentará una implementación de la comunicación entre los usuarios y la aplicación, ésta es una posibilidad dentro de las tantas existentes, dado que este primer nivel puede ser implementado con otras tecnologías. El propósito de esta sección es demostrar que es real y simple la implementación de la arquitectura propuesta. Por dicho motivo, no se proporcionarán detalles de la implementación sino que se expondrá una idea general y precisa de cómo se mapean cada uno de los componentes de este nivel a la tecnología utilizada.

La implementación es realizada con el framework Struts [46] que utiliza el lenguaje de programación Java, asimismo, este framework utiliza tecnologías como Servlet [47] y JSP [48]. Se decidió realizarlo con dicho framework por tres simples razones. Una de ellas, es que es un framework muy utilizado hoy en día en el desarrollo de aplicaciones Web. La segunda razón, es que utiliza el lenguaje de programación Java que es

Orientado a Objetos. Y la última razón, es que es una tecnología con la cual ya se han realizado pequeños trabajos de implementación lo cual provee una ventaja con respecto a su utilización.

La comunicación entre los componentes que va a ser implementada se encuentra representada en la Figura 5.4.

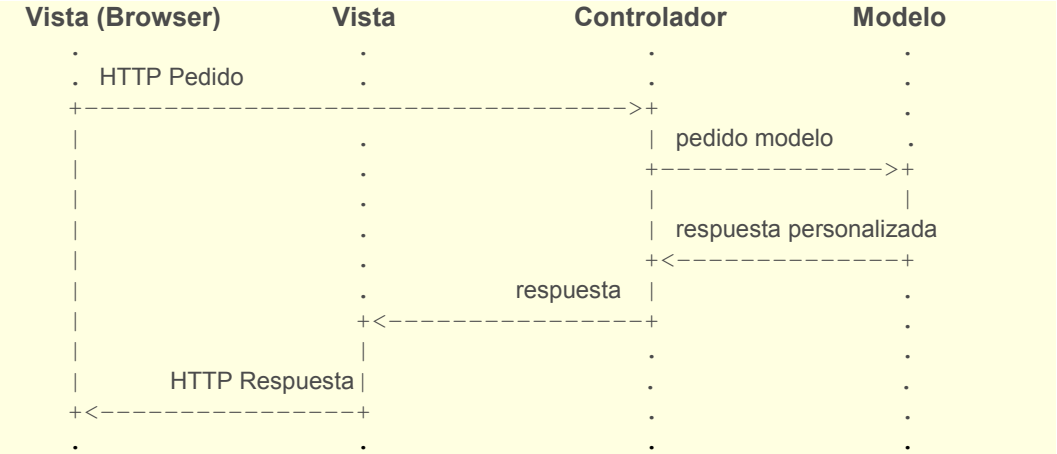

Figura 5.4: Diagrama de secuencia de una Petición.

Se explicará cómo son representados los componentes del MVC en la tecnología Struts y como es realizada la comunicación entre ellos. En este framework las funcionalidades son implementadas utilizando el lenguaje Java, por lo cual, se realizará la explicación utilizando términos de programación orientada a objetos, como por ejemplo, instancia, clases y métodos, entre otros.

La comunicación comienza a partir del usuario. Llega desde el browser un pedido realizado por el usuario, que es atendido por el Controlador. En Struts el Controlador es un conjunto de eventos que a través de clases Java mapean los pedidos de los usuarios (generalmente requerimientos HTTP) a métodos de clases. Un servlet llamado ActionServlet, que es provisto por el framework Struts, es el encargado de atender los pedidos realizados por todos los usuarios *(Controlador)*. Este ActionServlet es parte de una implementación del patrón Command [49] para lograr mapear los pedidos realizados por los usuarios a métodos entendibles por el modelo. Este ActionServlet crea un objeto Action, en caso de que este objeto haya sido creado anteriormente utiliza la instancia creada previamente. Cada Action que se instancia es una subclase de una jerarquía de Actions, donde existe una subclase por cada pedido que puede ser realizado por el usuario. Estas subclases Action son creadas de manera particular para cada aplicación que se implemente. El mapeo para saber qué objeto Action se debe instanciar se encuentra en un archivo llamado struts-config.xml, este xml contiene todos los mapeos de qué Action instanciar de acuerdo al pedido realizado por el usuario, en otras palabras configura el Command. El objetivo de los objetos Actions es el de mapear el requerimiento realizado por los usuarios a métodos del Modelo, por lo cual en cada una de las clases Action se encuentran los métodos del Modelo que deben ser ejecutados para un pedido particular. Estos Actions son parte del Controlador de la aplicación.

El proceso completo del Controlador es el siguiente, el ActionServlet recibe el pedido realizado por el usuario e instancia el Action correspondiente, logrando el Action correspondiente a través del archivo struts-config.xml, y este Action sabrá a qué objeto del Modelo debe llamar y qué método o métodos de este objeto se debe ejecutar. Como se explico anteriormente, la explicación de cómo se resuelve este pedido dentro del Modelo queda pendiente para el capítulo siguiente. Sin embargo, se puede adelantar que

el Modelo resuelve el pedido retornando un objeto que contiene los datos para resolver la respuesta personalizada. Este objeto respuesta es devuelto al Action que le realizó el pedido, esto sucede dado que el Action ejecuta un método execute que ejecuta métodos del Modelo, y al término de ellos retorna al Action la respuesta. El Action que ejecuta estos métodos contiene parámetros con valores necesarios para la ejecución correcta de los métodos. Además de los parámetros se posee la sesión de cada usuario, cuyo objetivo es el de tener un lugar reservado para el almacenamiento de la información del usuario y el almacenamiento temporal de objetos que se necesiten conservar mientras perdure la resolución del pedido realizado. Se puede pensar en este lugar como una bolsa de objetos que contiene una clave asociada a datos almacenados. Una vez que el Action obtiene la respuesta del Modelo, retorna el control con la respuesta al ActionServlet.

El ActionServlet una vez que le retornaron la repuesta al pedido que realizó, deriva el proceso de transformación de los datos para ser vistos por el usuario en una página JSP asignada a resolver el pedido requerido. Como propiedad de las páginas JSP, se puede destacar sin entrar en detalle que no son páginas estáticas, como HTML, sino que pueden ser formadas dinámicamente[15] [50], y además, poseen las características de realizar procesamiento de datos y manejar parámetros para poder construirlas dinámicamente, y lograr como por ejemplo una página HTML personalizada creada como resultado. Por lo cual, la página JSP utilizará los tags para la creación dinámica de sus componente y basándose en los valores que posee el objeto que es devuelto por el Modelo podrá armar la página de respuesta personalizada, con los componentes que el usuario desea ver, en el orden y con el color de preferencia del mismo.

La interacción completa de los componentes de Struts implementando este primer nivel puede observarse gráficamente para un mejor entendimiento en la Figura 5.5.

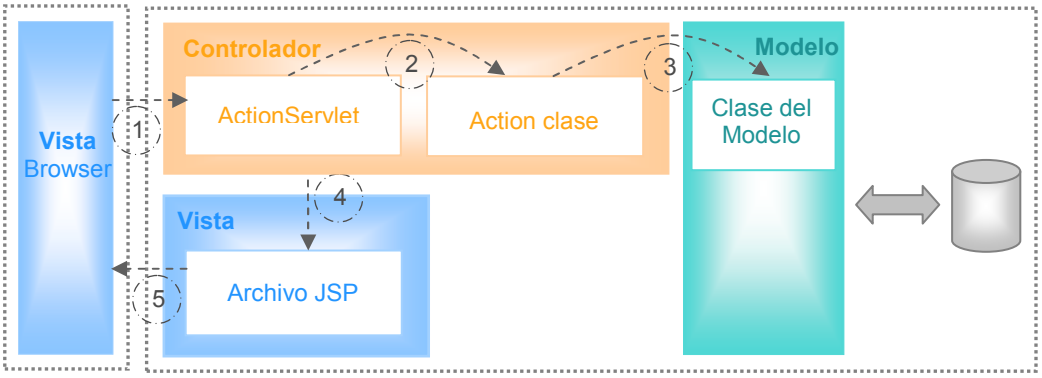

Figura 5.5: Interacción en el framework Struts.

Como puede observarse en esta sección el primer nivel de la arquitectura puede ser implementado de manera simple con el framework Struts. Se deben desarrollar las clases Action dependiendo de los pedidos que deban ser atendidos por la aplicación, y las páginas JSP para que puedan ser creadas de manera dinámica, quizás teniendo previamente el armado de encabezados y cuerpo de las páginas utilizando Tile, o JSP prearmadas.

15 Existe también Tile, que es un modo de utilizar templates para poseer páginas JSP prearmadas [51].

5.5 Conclusiones

En este capítulo se expuso como primer paso, una explicación de qué modo será expuesta la arquitectura propuesta, se expusieron las características generales de la misma y luego se presentó el primer nivel de abstracción de la arquitectura construido utilizando el patrón MVC. El desarrollo de este primer nivel incluye el lugar de ubicación de los componentes del patrón MVC entre el cliente y el servidor, la manera que se debe realizar el desarrollo de la aplicación utilizando el mismo, exhibiendo las responsabilidades de cada uno de los componentes que lo constituyen y cómo se realiza la comunicación entre ellos. Posteriormente, se presentó una posible implementación con el objetivo de mostrar con cuales de las tantas tecnologías existentes en la actualidad puede ser implementado este primer nivel.

Al realizar la descripción de la funcionalidad de cada uno de los componentes del MVC y explicación de cómo se realiza la comunicación entre ellos, se resaltaron los beneficios que aporta su separación de responsabilidades en su utilización como parte de la arquitectura propuesta. Sobre los mismos caben destacarse como principales ventajas las siguientes:

- Independencia del Modelo con respecto a los aspectos de interfaz.
- Se pueden realizar múltiples tipos de interfaz de usuario basándose en un único Modelo.
- Se puede manejar la multiplicidad de interfaces agregando o eliminando funcionalidad en el Controlador y en la Vista, sin modificar el Modelo.
- Modificaciones realizadas en el Modelo, como cambios en la lógica de la aplicación, dominio o en las reglas del negocio, no constituyen cambios en el Controlador o en la Vista.
- El controlador y la Vista no poseen aspectos de personalización. Esta funcionalidad se encuentra encapsulada en el Modelo.

Se puede resaltar como una característica la modificación que se puede observar en esta arquitectura con respecto al MVC tradicional, la utilización de un componente Vista que se encuentra dividido en dos partes. Con esta división se logra la independencia de la interfaz con respecto al Modelo, desligando por completo al Modelo de esta funcionalidad.

Otra característica importante a destacar en este primer nivel es con respecto a la personalización. Si bien el Modelo de la aplicación no es descripto con detalle en este capítulo, se explicó que dentro del mismo se encapsula todo el comportamiento que se encuentra ligado a la personalización en la aplicación. De este modo, se logra que se conserve la independencia de funcionalidad de los componentes que constituyen el MVC, logrando de esta manera conservar el objetivo principal del patrón MVC.

Finalmente se debe marcar una posible desventaja creada por la utilización de un patrón de diseño como lo es el MVC. Esta desventaja se debe a que la utilización del MVC requiere de una cierta capacidad y visión de importancia con respecto al diseño de las aplicaciones que no posee cualquier desarrollador hoy en día. Este patrón, como cualquier otro, implica invertir cierto tiempo destinado a pensar cómo va a ser desarrollado. Estas posibles desventajas no fueron tomadas en cuenta a la hora de decidir incluir este patrón como parte de la arquitectura. Esta decisión se debió a que este trabajo se encuentra orientado a lograr soportar aplicaciones Web personalizadas de gran envergadura y con un cierto grado de complejidad, y colocando estas desventajas en contraposición a las ventajas que su utilización aportan, no podrían ser tomadas en cuenta dado que son muchísimos más los beneficios que su utilización provee.

Capítulo 6

Modelo Interno de la Arquitectura

Este capítulo es uno de los más importantes de este trabajo dado que presenta el Modelo del MVC en el cual se encuentra toda la lógica de la aplicación y las funcionalidades para armar los nodos y realizar su personalización. Se presenta cómo se encuentra formado del modelo de la aplicación, qué elementos lo componen y cómo se desarrolla su interacción. Se desarrollan las diferentes responsabilidades de cada uno de los elementos que lo componen, cuáles aspectos personalizan cada uno de ellos, y cómo realizan su interacción con los perfiles de los usuarios para lograr la personalización de los nodos.

En la primera sección, sección 6.1, se realiza la descripción del Modelo, se indican los modelos que lo componen y sus características, y además, se describe el porqué de cada uno de los componentes que lo integran. Se comprende una subsección en donde se exhibe de dónde proviene la idea de la creación de cada uno de los modelos. En las secciones anteriores uno de los componentes del Modelo no es descripto, el Perfil de Usuario, al que se le dedica la sección 6.2, aquí se describe qué representa, qué función cumple y qué información es almacenada en él. Se introdujo una subsección que describe las posibilidades de ubicación del perfil entre el lado del servidor y del cliente, ventajas y desventajas de cada una de ellas, y cuál es la decisión tomada con respecto a la ubicación de los perfiles de usuario en esta arquitectura y el porqué. En la sección 6.3 se realiza una explicación de cómo son armados cada uno de los nodos para un mejor entendimiento cuando luego se presenta la funcionalidad de cada modelo. Además, se realiza una descripción general de cada uno de los modelos para a continuación describirlos en detalle en las subsecciones 6.3.1, 6.3.2 y 6.3.4. En la subsección 6.3.3 se presentan las reglas de negocio y su intervención en la arquitectura. Y como última introducción de los modelos, se encuentra la implementación de los mismos en la sección 6.4. Para concluir con este capítulo se presenta una sección de conclusiones, sección 6.5, para cerrar el capítulo destacando las características más importantes de este capítulo y las conclusiones hechas en base a las explicaciones que fueron brindadas sobre el modelo de la aplicación.

6.1 Diseño Interno del Modelo

Como se explicó en el capítulo anterior este componente es el más complejo y el que posee más importancia en este trabajo, esto sucede dado que encapsula la lógica de la aplicación junto con los aspectos de personalización que la aplicación brinda, además como esta arquitectura fue realizada con el objetivo de soportar aplicaciones Web de comercio electrónico personalizadas, este componente también contiene las reglas de negocio.

Antes de comenzar a explicar el Modelo de la aplicación se debe destacar un concepto que será muy utilizado a lo largo de este capítulo, el concepto *nodo*. Nodo se define en el término de las aplicaciones hipermediales, y en OOHDM se encuentra definido como una vista sobre las clases conceptuales [15], estas clases representan las clases del domino de la aplicación y dichos nodos son definidos durante el tiempo de diseño. Estos nodos contendrán diferentes atributos obtenidos de diversas clases conceptuales. Además poseen aspectos de navegación como son los links, a los que se puede acceder mediante los mismos nodos y navegar a través de ellos hacia otro nodo con diferente información y características. Una vez explicado el concepto de nodo se continuará con la presentación del Modelo.

El objetivo del componente Modelo es el de lograr armar un nodo con la información correspondiente al pedido realizado, donde dicha información se encuentra personalizada en base a las preferencias del usuario. Este componente tiene la responsabilidad de resolver los pedidos realizados por los usuarios en base a los datos que forman la funcionalidad de la aplicación, así como también, aplicándoles a estos datos procesamientos para poder personalizar las respuestas que va a ser enviadas a los usuarios según sus preferencias y gustos. El Modelo se encuentra compuesto por la lógica de la aplicación, su dominio, las reglas de negocio y los aspectos de personalización. Para conseguir alcanzar este objetivo, este componente debe interactuar con la base de datos para obtener y persistir la información adecuada para procesar los pedidos y crear el nodo personalizado resultado de la solicitud realizada por el usuario.

La definición lograda de este Modelo se obtuvo en base a conceptos definidos en la metodología OOHDM. Este Modelo se encuentra compuesto por cuatro módulos, cada uno cumpliendo con una responsabilidad diferente. Los cuatro módulos que lo constituyen son el Modelo Navegacional, el Modelo de Aplicación, el Modelo de Interfaz y por último el Perfil de Usuario. El módulo Perfil de Usuario en realidad representa muchos perfiles de usuarios, pero se nombra de esta manera para tener un concepto general de los mismos. El objetivo de esta división de conceptos y funcionalidad entre los módulos se debe al propósito que se fue manifestando a lo largo de los últimos capítulos, los beneficios que aporta la separación de responsabilidades, tales como lograr independencia en el diseño y desarrollo de la aplicación, hacer posible su mantenimiento y futuras modificaciones, y además logrando que estas dos últimas tareas puedan ser realizadas de una manera sencilla y fácil, intentando que se deban modificar los módulos adecuados para ello sin que impliquen cambios drásticos en el resto de la aplicación. Manteniendo estos propósitos como principales objetivos se generó este Modelo compuesto de cuatro módulos.

El gráfico de los componentes que lo formar y cómo es su ubicación en la arquitectura se observa en la Figura 6.1.

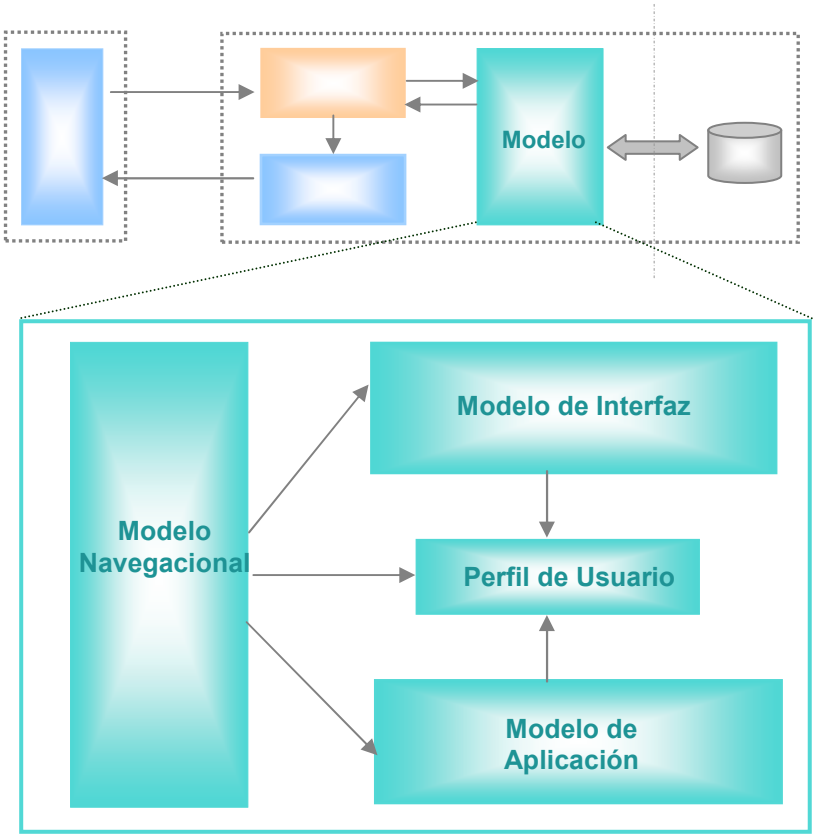

Figura 6.1: Modelo interno de la arquitectura.

La realización de este Modelo se encuentra pensada en base a la utilización de la Programación Orientada a Objetos [36], por lo cual los módulos que lo componen se encuentran representados por clases. Se efectúa esta aclaración para que la interpretación de las explicaciones siguientes tenga una visión más concreta y se logre un mejor entendimiento.

En las secciones siguientes se expondrán las características y funcionalidades de cada uno de los módulos que componen el Modelo, así como también como se realiza la comunicación entre ellos. La interacción que este Modelo realiza con el primer nivel de la arquitectura, descripto en al capítulo anterior, queda explicado a lo largo de este capítulo.

6.1.1. ¿Cómo surgen los módulos?

Antes de comenzar dando una explicación de cada uno de los módulos, parece apropiado exponer una idea general del rol que cumplen cada uno de los módulos y cual es el origen de cada uno de ellos, y de este modo entender mejor la explicación de las siguientes secciones en donde se profundizan cada uno de dichos módulos.

La división que se propone de los diferentes módulos y su funcionalidad surgen en base a los conceptos aportados por la metodología OOHDM y en las características estudiadas que poseen las aplicaciones personalizadas ligadas a los aspectos de las aplicaciones que pueden ser personalizados.

Cómo primera instancia se puede nombrar el Perfil de Usuario, en realidad este módulo no contiene un solo perfil como su nombre en singular así lo indica, este módulo representa todos los perfiles de usuarios que contiene la aplicación. Este módulo es fundamental y necesario dado que la arquitectura soporta aplicaciones Web personalizadas, y la única manera que posee de personalizarla es almacenando datos de

los usuarios con sus preferencias. Por dicha razón surge la creación del módulo Perfil de Usuario. El Modelo Navegacional en cambio, surge del concepto del Diseño Navegacional de OOHDM. En uno de los pasos de diseño, se desarrolla el diseño de la navegación de la aplicación conteniendo el diseño de los nodos que son vistas de los atributos de los objetos del dominio de la aplicación, sumándole a dichos nodos aspectos de navegación. Este módulo será el encargado de los nodos navegacionales, armándolos de manera personalizada. Luego el Modelo de Aplicación es el que contiene la lógica propia de la aplicación así como los objetos del dominio, y en esta arquitectura por ser orientada a aplicaciones de comercio electrónico, posee las reglas del negocio. Este módulo es el corazón propio de la aplicación, y además de ser totalmente necesario poseerlo para resolver características propias de la aplicación. Se deriva del Diseño del Modelo Conceptual de OOHDM, el cual diseña los objetos del dominio de la aplicación. Por último, el Modelo de Interfaz surge a partir del Diseño de Interfaz Abstracta (ADV), que define la interfaz de la aplicación en términos abstractos. Este módulo no tiene la misma finalidad que posee el Diseño de Interfaz en OOHDM, dado que en la arquitectura propuesta este módulo se encarga de los aspectos de interfaz como los colores, posición de los elementos de interfaz, en cambio en OOHDM, estos aspectos no son parte de las ADV's.

Puesto que la arquitectura propuesta deberá resolver aplicaciones Web personalizadas, es necesario que la aplicación sea capaz de construir las vistas dependiendo del perfil de usuario, es decir, en forma totalmente dinámica, por ello la arquitectura contiene dichos módulos que se encargarán de personalizar cada una de las funcionalidades y un módulo que representa a los perfiles de usuario que es el encargado de almacenar la información de los usuarios. Como se describió anteriormente, con los aportes de la metodología OOHDM y las características que deben ser agregadas para realizar la personalización surgieron cada uno de los módulos, logrando de este modo, distribuir las responsabilidades de las partes para obtener un mejor diseño de la aplicación y desacoplamiento de la funcionalidad.

6.2 Perfil de Usuario

Las aplicaciones personalizadas de alguna manera deben manejar y almacenar datos de los diferentes usuarios para lograr realizar la personalización. Para lograr el almacenamiento de dichos datos se deben realizar los siguientes pasos: recolección de datos de los usuarios, modelar y categorizar estos datos a ser almacenados (proceso realizado en el diseño de la aplicación), el análisis de los datos que son recolectados, y la determinación de las acciones a realizar. Esta arquitectura como soporta aplicaciones Web personalizadas debe poseer procesamiento y almacenamiento para dichos datos. Para lograr esto, surge el módulo Perfil de Usuario cuya función es la de realizar el almacenamiento de los datos de los usuarios. Este módulo se representa con dicho nombre pero en realidad, se debe pensar en dicho módulo como un conjunto compuesto por todos los perfiles de los usuarios que acceden a la aplicación.

Para efectuar el manejo de todos los perfiles de usuarios que contendrá la aplicación se debe poseer en la misma un manejador que será el responsable de administrarlos. Este manejador será el encargado de realizar las tareas de creación, eliminación, actualización y modificación de los perfiles de usuarios. Además, una de las tareas fundamentales del mismo es la responsabilidad de encontrar entre todos sus perfiles el perfil del usuario que solicitó el pedido que se está procesando para que los módulos interactúen con él, y de este modo, poder personalizar la información que será enviada como respuesta a dicho usuario. Para llevar a cabo esta tarea de una manera más eficiente quizás sea necesario utilizar algún mecanismo de orden para lograr encontrar los perfiles requeridos en el menor tiempo posible, pero esta determinación depende fundamentalmente de los datos a

tener en cuenta para reconocer a los usuarios, que pueden variar de aplicación en aplicación.

Cada perfil de usuario contenido en la arquitectura se encuentra formado como se describió en la sección 2.4.2, con datos propios del usuarios, como nombre y apellido, edad, lugar de residencia, entre otros y con datos orientados a almacenar preferencias de aspectos de la personalización, como los son los colores de la interfaz, posiciones de los elementos de interfaz, características de intereses en cuanto a las búsquedas, la manera de pago, entre otras. Los valores especificados para un perfil de usuario son determinados por el conjunto de atributos definidos y los valores de cada atributo. Por lo cual, los modelos que componen al componente Modelo obtienen información solicitándosela al perfil del usuario que realizó el pedido para lograr personalizar cada uno de los aspectos que forman los nodos y obteniendo una respuesta personalizada para el usuario. Como se puede observar en la Figura 6.1, cada uno de los módulos interactúa con el perfil para personalizar el aspecto del cual es encargado el módulo, pidiéndole al perfil la información necesaria para cumplir dicho objetivo. El perfil juega un rol más pasivo comparado con los módulos dado que no requiere interactuar con ellos, solo les brinda la información que ellos le solicitan.

La información que va a ser almacenada en el perfil de usuario dependerá de los aspectos de la aplicación que se personalicen. Puede suceder que se personalice el contenido de la información, y no se personalice la interfaz, para lo cual será necesario almacenar preferencias en cuanto a los intereses del usuario, pero no se almacenará ningún dato con respecto a preferencia de colores u otro aspecto que se encuentre relacionado con la interfaz. Por lo cual, la cantidad y propiedades de los atributos dependerán de la información que sea necesaria almacenar para lograr personalizar los aspectos de la aplicación a ser personalizados. El diseño de los atributos que van a ser almacenados en los perfiles pueden variar entre las diferentes aplicaciones. En la fase de diseño de la aplicación, un paso que debe realizar el diseñador es definir los ítems de información que serán almacenados en los perfiles de usuarios, teniendo en cuenta qué información va a ser necesaria almacenar y cómo se va a organizar para resolver los aspectos de la aplicación que son personalizados. También se debe tener en cuenta que cierta información que será almacenada en el perfil es fundamental para realizar la tarea de identificación de los usuarios.

El desarrollo de los perfiles durante el tiempo de vida de la aplicación se da de la siguiente manera. Los perfiles de usuarios se irán creando dentro de la aplicación cuando sea necesario hacerlo, no previamente. Cuando un usuario accede por primera vez a la aplicación, no existe un perfil de usuario para él. Por lo cual, dicho perfil debe ser creado una vez que se realice la registración del usuario, almacenando en el perfil los datos que puedan obtenerse de dicha registración y de la interacción que el mismo efectúe con la aplicación. En caso de que no se utilice el login como medio de identificación de los usuarios, la primera vez que el usuario acceda a la aplicación se crea un perfil para él y además un archivo cookie es depositado en la máquina del cliente para su reconocimiento cuando se realice un próximo acceso a la aplicación. Una vez que el perfil es creado, el mismo se irá actualizando y conteniendo más información mediante las interacciones que el usuario realice en nuevos accesos a la aplicación, obteniendo de esta manera más información del usuario, y de este modo mejorando la personalización de la aplicación.

Se deberá utilizar, en caso de que se requiera, una política de eliminación de perfiles de usuarios. Esta política se usará en caso de que se quieran eliminar los usuarios que han realizado la registración y no produjeron accesos a la aplicación por largo tiempo. Dichas políticas serán particulares de cada aplicación.

Como cierre de esta sección se puede destacar que como muestra la Figura 6.1 se tomó la decisión de colocar los perfiles de usuarios en el servidor, esta característica y las razones por las cuales se tomó esta decisión se detallan en la siguiente sección.

6.2.1. Ubicación del Perfil (servidor vs. cliente)

La decisión de ubicar los perfiles de los usuarios del lado del servidor en la aplicación no fue una decisión tomada al azar. Para realizarla se basó la determinación en las ventajas y desventajas que proveen las tres maneras de efectuar y procesar los datos de los perfiles [4]. Con la idea de presentar las opciones posibles se presentarán cada una de ellas con sus ventajas y desventajas, y se expondrán las razones de la decisión tomada.

La primera opción es ubicarlo del lado del cliente realizando el manejo del mismo del lado del servidor como muestra la Figura 6.2.a. En este caso, se almacena el perfil en el cliente y el procesamiento del mismo se efectúa en el servidor. Por dicha separación, el perfil de usuario debe viajar por la red una vez que el usuario accede a la aplicación para que pueda ser utilizado por la misma para personalizar los resultados a los pedidos solicitados. Requiere de cierto almacenamiento y un pequeño procesamiento para el manejo del perfil en el usuario. No obstante, posee la ventaja que puede ser compartido por otras aplicaciones o servidores utilizando la información que ya se encuentra almacenada, y con esto, provee la posible utilización de las técnicas de personalización, content-based filtering[16] y profile sharing[17] pero no provee collaborative filtering[18] dado que los perfiles se encuentran almacenados de manera distribuida. Una gran ventaja que proporciona es que el proveedor de la aplicación no debe disponer ni de software ni de espacio para almacenar estos datos ya que esto es realizado por cada usuario.

Otra opción, es que sea almacenado del lado del cliente y también procesado allí, obsérvese la Figura 6.2.b. Como la opción anterior, por encontrarse los perfiles distribuidos en las máquinas de cada usuario esta opción provee content-based filtering y profile sharing pero no provee collaborative filtering. Una de las grandes ventajas de ubicar el perfil del lado del cliente es tener una especie de *perfil abierto* que proporciona que otras aplicaciones puedan acceder al mismo el cual ya está creado y posee información del usuario, y de esta manera, se puede personalizar ni bien se accede al mismo. Con este modo los usuarios no tienen que proveer su información a cada aplicación que visiten ya que se completan sus datos y preferencias una sola vez, si bien después se va actualizando [52]. La desventaja de esto es que todas las aplicaciones deben conocer como comunicarse con el perfil, y además, cada una debe necesitar quizás información que el perfil no contempla.

La última opción expuesta en la Figura 6.2.c, es ubicar el perfil del lado del servidor y realizar su procesamiento ahí mismo, como sucede en la arquitectura propuesta. Este caso es eficiente ya que el perfil del usuario no necesita ser transmitido por la red cuando se accede a la aplicación. Además, la centralización de todos los perfiles de usuarios en un lugar genera la opción de poder utilizar las técnicas de content-based y collaborative filtering. Otra ventaja que provee es que no requiere que el usuario disponga de ciertos recursos, dado que las tareas de almacenamiento y procesamiento son llevadas a cabo en el servidor. La desventaja que esta opción provee es que el perfil es propio de la aplicación y no puede ser compartido con otras aplicaciones o diferentes servidores.

---

[16] Esta técnica se basa en las preferencias individuales del usuario. Consiste en recomendarles a los usuarios ítems basándose en el historial del mismo buscando ítems accedidos en el pasado.

[17] Esta técnica consiste en compartir un mismo perfil de usuario con diferentes aplicaciones.

[18] Esta técnica necesita que la aplicación posea acceso a todos los perfiles de usuarios. Se basa en la suposición de que usuarios con comportamientos similares poseen intereses análogos.

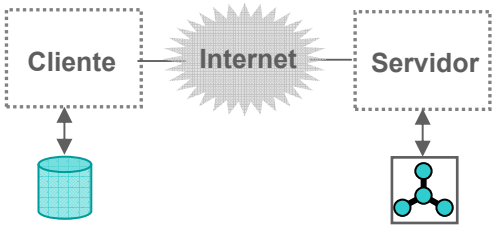

Figura 6.2.a: Almacenamiento en el cliente y procesamiento en el servidor.

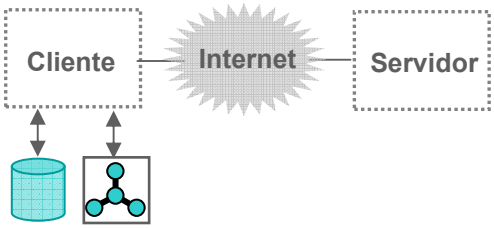

Figura 6.2.b: Almacenamiento y procesamiento en el cliente.

Procesamiento de los perfiles de Usuarios.

Almacenamiento de los perfiles de Usuarios.

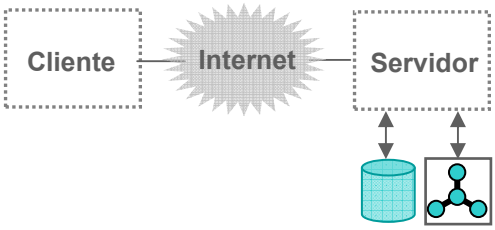

Figura 6.2.c: Almacenamiento y procesamiento en el servidor.

Las soluciones que colocan el perfil en el cliente requieren de un cliente Web con mayores requerimientos, es decir que en el cliente se debe ubicar una aplicación especial para manejarlo, además del almacenamiento y el tráfico de red que provoca la interacción con el perfil de usuario para resolver la personalización.

La solución que incluye el perfil de usuario y su procesamiento en el servidor, contiene la ventaja de que todo el procesamiento e interacción entre el perfil y la lógica de la aplicación se realiza en el servidor, lo que no implica tráfico de red, y por lo cual, el tiempo de respuesta es menor, se gana en performance, y otra ventaja es que el cliente no posee ningún procesamiento perteneciente a la aplicación. La desventaja que esto ocasiona es que el usuario cuando accede a la aplicación necesita de un mecanismo de identificación para que la aplicación lo reconozca y así pueda interactuar con el mismo.

Teniendo en cuenta las ventajas y desventajas que poseen cada una de las opciones y sobre todo la división realizada de los módulos que componen el Modelo de la arquitectura, se decidió ubicar los perfiles de usuario y su procesamiento en el servidor. De esta manera, cada módulo cuando deba interactuar con el perfil posee un mínimo tiempo de latencia que se debe a la búsqueda de obtener el usuario correspondiente de entre todos los perfiles almacenados. Además, no son utilizados recursos del usuario para la realización de la personalización y con esta opción se pueden realizar las técnicas content-based y collaborative filtering.

6.3 Módulos que componen el Modelo

Para lograr una explicación comprensible de los modelos de la arquitectura se brinda en esta sección una breve exposición de como se encuentran formados los nodos de la aplicación a ser presentados a los usuarios, que son llamados de una manera más común en el lenguaje Web, páginas Web. Al término de esta sección se expondrá una pequeña descripción del Modelo en general.

El espacio navegacional de las aplicaciones es visto como un conjunto de nodos que se van accediendo de unos a otros a través de los links que poseen los mismos. El objetivo final de la arquitectura es lograr armar nodos personalizados para que los usuarios accedan a la aplicación a través de ellos. Los nodos de la aplicación, como se

explica en la metodología OOHDM [15], son diseñados antes del desarrollo de la misma, sabiendo de antemano como está compuesto cada uno de ellos, y pensando en ellos como vistas de atributos de los objetos conceptuales de la aplicación, o sea, exponiendo atributos de los objetos del dominio. En el caso de las aplicaciones personalizadas, si bien el contenido de cada uno de los nodos se encuentra previamente decidido, varía con los intereses y preferencias de cada uno de los usuarios, pero la base de cada nodo no cambia, son sus aspectos los que son adaptados a cada usuario. En este trabajo se tratarán los nodos como elementos que se encuentran compuestos por bloques de información, teniendo cada uno de ellos predefinidos los bloques que contienen. Por lo cual teniendo predefinidos los nodos, dichos bloques de información que los componen serán adaptados con los gustos e intereses de los usuarios, así como por ejemplo se adapta su contenido y colores entre otros aspectos que pueden ser personalizados.

Para quede expresada un poco más clara esta idea de los bloques de información se presenta en la Figura 6.3 un ejemplo del nodo de MyYahoo y se muestra como queda dicho nodo representado con bloques de información.

Figura 6.3: Bloques de información que contiene el nodo.

No existe un pasaje pautado de como debe estar compuesto un nodo por los bloques de información, en realidad la idea de los bloques de información dependerá del contenido de información de cada uno de los nodo y cómo se quiera representar el nodo en la aplicación para poder ser llevada a cabo su construcción.

Esta idea presentada donde los nodos navegacionales se encuentran formados por bloques de información, va a ser la base de la explicación de cómo en la arquitectura se van a ir construyendo los nodos. En las secciones siguientes se continúa con la exposición de cómo estos bloques de información son generados por la arquitectura propuesta, y como se va a personalizar su contenido, estructura y visualización, para lograr el nodo personalizado de respuesta.

Como se observa en la Figura 6.1 el Modelo del MVC está compuesto por el Perfil de Usuario, y tres modelos más que interactúan con el anterior. Esta división se debe por el motivo que la arquitectura presenta la idea de que los nodos van a ser creados por partes, incluyendo cada uno de estos modelos un poco más de información hasta completarlo y obtener el nodo completo y personalizado a ser exhibido al usuario que lo

solicitó. A continuación se presentará cada modelo que compone el Modelo del MVC con sus características y sus responsabilidades. Ya que cada uno de ellos es encargado de armar solo una fracción del nodo, se expondrá cuál es su función, cómo interactúan con el perfil para lograr la personalización y con ello obtener la información necesaria para cumplir con su tarea, y cómo es pasada la responsabilidad de la creación del nodo de un modelo a otro para ir completándolo y obtener el nodo final.

Un último asunto a destacar antes de comenzar con la explicación de los modelos, que parece correcto remarcar es, que si bien la explicación de las funciones de los modelos se realiza sin pensar y sin tener en cuenta la implementación, se debe pensar que cuando se hable de ellos se refieren a clases y métodos de la programación orientada a objetos.

6.3.1. Modelo Navegacional

Una de las funciones importantes que el Modelo Navegacional realiza, es la interacción del segundo nivel de la arquitectura con el primer nivel de la misma, descripto en el capítulo 5. Este modelo es el encargado de recibir los pedidos que el controlador solicita al Modelo del MVC, y luego realiza la tarea de enviar, una vez generado, el nodo de respuesta correspondiente al Controlador.

Como se explicó anteriormente, cada modelo que compone al Modelo del MVC va cumpliendo con una tarea para lograr en conjunto el resultado personalizado al pedido hecho por el usuario. Este Modelo Navegacional posee la responsabilidad de la creación de los nodos, y en cuanto a lo que personalización respecta, es el encargado de los aspectos de la personalización en cuanto a la estructura del nodo. O sea, cuando la personalización de la aplicación incluye la selección o exclusión de bloques de información en un nodo determinado y la estructura del nodo puede variar de un usuario a otro. Estas tareas son realizadas de la manera siguiente.

El objetivo de la arquitectura es el de realizar la creación de los nodos navegacionales personalizados como respuestas a los pedidos realizados por los usuarios, y dentro del Modelo, este pedido comienza por el Modelo Navegacional. Este modelo como primera tarea de creación de los nodos se encarga de construir los moldes de los nodos navegacionales, incluyendo en los mismos los bloques de información que dichos nodos van a contener. Esto sucede de este modo dado que, como se explicó en la sección anterior, se sabe de antemano como se encuentran formados los nodos, por lo cual se sabe qué bloques de información lo componen. Además, como fue expuesto en la sección 5.3.1, el Controlador posee la funcionalidad de traducir los pedidos realizados desde la vista a mensajes entendibles por el Modelo. Este Controlador sabe a qué clase del Modelo Navegacional debe realizarle el pedido. Por lo cual, el Controlador al efectuarle el pedido a la clase que corresponde en el modelo navegacional, la misma sabrá armar el nodo correspondiente con los bloques de información que deben ser incluidos en el mismo. Realizando solo esta tarea se posee un nodo de navegación sin personalizar, aquí es donde el perfil de usuario es incluido en el proceso para lograr la personalización de la estructura de dicho nodo. Con el pedido que efectúa el Controlador viene como parámetro el usuario que lo realizó, y con esta información se logra identificarlo. Cuando se personaliza el aspecto de estructura en la aplicación, antes de armar el nodo la clase responsable debe pedirle al manejador de perfiles el perfil que corresponde a dicho usuario, y de este modo interactuando con el mismo, sabrá qué bloques de información deben ser incluidos y cuales no. Este aspecto personalizado puede observarse en el sitio personalizado MyYahoo, en donde cada usuario tiene la posibilidad de seleccionar qué bloque de información desea visualizar y cuales no son de su interés. Una posibilidad de implementación es que el perfil de cada usuario puede poseer una colección de bloques de información que desea que sean incluidos en el nodo que se le mostrará, y así el nodo agregará solo los bloques que dicho usuario tiene interés de ver y se resuelve de este modo la personalización de la estructura, obteniendo el

molde del nodo a construir. Dentro de dicho molde se encontrarán los bloques de información que forman el nodo, y así se sabrá qué información debe ser pedida al Modelo de Aplicación y qué datos de la interfaz corresponden a dichos bloques que deben ser completados. Otra opción que se presenta con este tipo de personalización es en el caso que se presente diferente información dependiendo del tipo de usuario que se encuentra accediendo. Por ejemplo, en el caso de que existen categorías de usuarios como clientes gold, silver, u otros, cuando el usuario se registra y el pedido ya se encuentra en al Modelo Navegacional, accediendo a la clase con el nodo que se desea ver. Dicha clase interactuará con el perfil del usuario el cual sabrá qué categoría de cliente es, y con dicha información e interactuando con clases que tengan como conocimiento los bloques de información a los cuales debe acceder cada cliente de cada categoría, se obtendrán de esta manera los bloques de información que tendrá dicho nodo para el tipo de usuario que accedió. Esta puede ser otra manera en que se presente la personalización de la estructura, en tal caso, quizás se tenga una jerarquía de subclases con el objetivo de categorizar a los usuarios incluyendo en cada una de ellas una colección con los bloques de información que deben integrar el nodo.

Una pregunta que puede llegar a surgir es, ¿qué pasa si el usuario accede a la aplicación y el perfil del usuario no existe? Este caso sucede solo cuando los usuarios acceden por primera vez a la aplicación, dado que, sino su perfil ya habría sido creado anteriormente. Si la aplicación utilizara cookies para el reconocimiento de usuarios, al no poder identificarlo, el usuario es derivado a una página de registración, nodo de registración. En caso que la aplicación no utilice esta tecnología, cuando el usuario quiere acceder a la aplicación se requiere que se registre o que ingrese sus datos, como es la primera vez que accede, se realiza la registración y se comienza con la obtención de datos del usuario. En este momento, es cuando la aplicación debe crear un perfil de usuario para el nuevo cliente. Luego de aceptar los datos ingresados en la registración, dentro de la aplicación comienza el pedido de dirigirse al nodo de bienvenida y además se debe crear un perfil de usuario nuevo. Este pedido que fue realizado al Controlador es enviado luego al Modelo Navegacional, el cual tiene la responsabilidad de interactuar con el manejador de perfiles pedirle que cree un usuario nuevo y con los datos que le envía como parámetro se comienzan a completar los datos del mismo.

El Modelo Navegacional además de las tareas anteriores es el que centraliza el armado de los nodos, enviando cuando corresponda el molde de los nodos a los demás modelos para que ellos continúen con las tareas que deben realizar sobre los mismos. Se puede observar esta relación en la Figura 6.1 expuesta en la sección 6.1. Por esta razón, una vez que el Modelo Navegacional termina las tareas que debe realizar para armar el molde del nodo, pasa la responsabilidad de seguir trabajando sobre él al Modelo de Aplicación. La respuesta enviada del Modelo de Aplicación le es devuelta y se encarga de enviarla al Modelo de Interfaz, para que realizando sus tareas complete el nodo y éste sea enviado como respuesta al Controlador.

6.3.2. Modelo de Aplicación

El Modelo Navegacional una vez que concluyó con el armado del molde del nodo lo envía al Modelo de Aplicación. La tarea de éste consiste en completar la información referente al contenido en cuanto al texto que van a poseer los nodos, interactuando con el perfil del usuario para personalizar dicha información.

Este modelo representa el corazón de la aplicación. Se encuentra abstraído totalmente de los aspectos de interfaz de la aplicación, solo contiene el dominio de la misma. Está compuesto por las entidades del dominio de la aplicación, por la lógica de la misma, que se encarga de describir aspectos relacionados con la aplicación que soporta la lógica de negocios, incluyendo además las reglas de negocio, dado que la arquitectura se encuentra orientada a las aplicaciones de comercio electrónico.

Al Modelo de Aplicación se le envía el molde del nodo que se está armando, una vez que éste lo recibe debe completar cada bloque de información con el contenido que corresponda a dicho bloque. Solo se completa en este modelo el contenido de los bloques de información, interactuando con los elementos del dominio y con las reglas de negocio obteniendo la información necesaria para completarlos. Para lograr la personalización del contenido de la información, el modelo se comunica con el manejador de perfiles y le solicita el perfil del usuario que va a personalizar. Interactuando con el mismo se obtienen datos del usuario para personalizar su contenido. Por ejemplo, los algoritmos de recomendaciones de productos son incluidos en este modelo, los cuales tienen como objetivo recomendar productos que sean de interés para el usuario. Interactuando con el perfil del mismo, obtienen sus preferencias y productos de interés, logran obtener en base a dicha información productos que serán atractivos para el usuario. Además, como los usuarios se encuentran centralizados en el servidor para las recomendaciones puede ser utilizada la técnica de personalización collaborative filtering. Otro ejemplo más sencillo que se puede exponer con respecto al contenido, es en el caso de que la aplicación muestre el horóscopo de cada usuario. Para este caso el Modelo de Aplicación interactuará con el perfil obteniendo la fecha de nacimiento del mismo, y basándose en ella mostrará como contenido el horóscopo apropiado. También se manejan en este modelo los descuentos realizados a los usuarios que son clientes frecuentes. Esto se realiza interactuando con las reglas de negocio y las mismas con el perfil de usuario, obteniendo información de las últimas compras realizadas por el mismo, y de esta manera, se evalúa si le corresponde o no realizarle un descuento personalizando el precio de los productos para los diferentes usuarios.

Estos últimos son algunos de los ejemplos de personalización de contenido que pueden realizarse en las aplicaciones de comercio electrónico, y la forma en que son resueltos dentro de la arquitectura propuesta. Como puede observarse, para resolverlos siempre se interactúa con elementos pertenecientes al dominio de la aplicación en interacción con el perfil de usuario para obtener las propiedades del mismo y lograr la personalización adecuada.

Otro aspecto del contenido a señalar es la personalización de los links, si bien pueden ser tomados como parte del Modelo Navegacional se decidió resolverlos como contenido, o sea, en este modelo. Para resolverlos se obtendrá, realizando la interacción antes expuesta, el texto que se incluirá en el bloque de información como representación de los mismos, e información sobre el nodo al cual se dirige para que de esta manera pueda ser resuelto cuando el usuario realice el acceso a dicho nodo.

Si la aplicación careciera de personalización sobre el aspecto del contenido de la información, este modelo no efectuaría la interacción con el perfil del usuario y solo se interactuaría entre las entidades propias del domino de la aplicación para resolver el contenido de los nodos. Puede destacarse además, que en caso de que la aplicación no sea de comercio electrónico, las reglas de negocio no serían incluidas en este modelo.

Una vez concluida la tarea de completar el contenido de cada bloque de información que conforma el nodo, el Modelo de Aplicación retorna el molde del nodo un poco más completo al Modelo Navegacional para que éste posteriormente sea enviado al Modelo de Interfaz.

6.3.3. Reglas de Negocio

Las reglas de negocio representan las políticas y puede decirse además, las estrategias de negocio que posee la aplicación para aplicar a las operaciones que realizan los clientes. Se pueden nombrar como ejemplo, las políticas de descuentos realizados a los usuarios, las ofertas a realizarles, los diferentes procesos de check out para los usuarios, entre otras reglas que pueden ser aplicadas. Como se encuentran muy relacionadas con las entidades propias del dominio de la aplicación, dado que las reglas

se vinculan directamente con dichas entidades y deben interactuar con ellas para ser resueltas, son ubicadas en el Modelo de Aplicación como parte del mismo.

Las reglas de negocio son representadas como entidades dentro del Modelo de Aplicación. Dichas reglas sufren constantes cambios por su propia naturaleza, dado que las políticas propuestas por los negocios varían con el tiempo y la situación corriente del negocio. Estás van siendo adaptadas a las necesidades y tácticas que el negocio pone en marcha para atraer, mantener y brindar servicios a sus clientes. Pueden lograr sus objetivos obteniendo información de los objetos de dominio que poseen la información de los productos. Cada una de estas reglas es representada en la aplicación como una entidad que ejecuta una regla de negocio, además, dicha regla puede estar compuesta por varias entidades [53]. Las reglas tendrán conocimiento de con qué entidades del dominio deben interactuar para resolverse y obtener el contenido adecuado que será presentado al usuario. Asimismo, en las aplicaciones personalizadas dichas reglas pueden ser afectadas por la personalización [25] y necesitar adaptar su contenido al usuario que realizó el pedido. Por este motivo, para ser resueltas deberán interactuar con el manejador de usuario que les devolverá el perfil de usuario para que interactuando con él obtengan el contenido personalizado resultado de aplicar la regla de negocio.

Cuando se efectúa el pedido realizado por el usuario al Modelo de Aplicación se interactuará con las entidades del dominio para resolverlos, pero éstas además deben ejecutar las reglas de negocio que afecten al contenido que irá en el nodo correspondiente. Estas reglas pueden o no modificar el contenido, sin embargo, deben ser ejecutadas para aplicar las políticas del negocio al contenido que puede ser afectado por ellas. Puede suceder el caso en que para un mismo contenido de información puedan ser aplicadas más de una regla de negocio, y como son desarrolladas una por una en la aplicación se brinde la posibilidad de que en tiempo de ejecución sean aplicadas sobre un mismo contenido a mostrar, como una combinación de reglas como se expresa en [53].

Este desarrollo propuesto de implementar las reglas de negocio como entidades a parte de las entidades del dominio sustenta la idea principal de este trabajo, la separación de responsabilidades aportando amplios beneficios en este tipo de aplicaciones. Uno de los beneficios más importantes es que como son modificadas frecuentemente al ser independientes del resto del aplicación no se extienden las consecuencias de sus modificaciones, sí puede llegar a verse afectadas otras entidades en algunos casos particulares.

Como se explicó en secciones anteriores, este tema no es desarrollado en profundidad, sin embargo es tratado por la razón que se quiere dejar registro de dónde deben ser ubicadas las reglas dentro de la arquitectura, y además, como es su participación en el desenvolvimiento de la personalización.

6.3.4. Modelo de Interfaz

En el momento que el Modelo de Aplicación retorna la respuesta al Modelo Navegacional, para completar el nodo la única información que hace falta es la información visual del mismo. Para lograr dicha información existe el Modelo de Interfaz, al cual el Modelo Navegacional envía el nodo casi completo para finalizar su creación. Este modelo posee la tarea de completar los datos del nodo con respecto a las características de su visualización. Una vez que el nodo llega a este modelo, sus datos están casi completos y ya se sabe qué bloques de información va a contener, solo se necesita obtener la información de la visualización del mismo, colores, fuentes y posiciones que tendrán dichos bloques. Para ello, el Modelo de Interfaz pide al manejador de perfiles el perfil del usuario correspondiente e interactúa con el mismo para obtener los datos que precisa. Para lograr completar dicha tarea el perfil requiere contar con toda la información necesaria para terminar de armar el nodo adecuándolo a las preferencias del usuario.

Con respecto a la visualización del nodo existen varios aspectos que pueden ser personalizados, en ellos se incluyen, el color de fondo, el color del texto, la fuente que es utilizada para el texto, el tamaño de la misma, entre otras más posibilidades. Otro aspecto importante que se presenta en algunos sitios personalizados, es la capacidad de poder personalizar en qué lugar del nodo van a ser ubicados los bloques de información que lo forman. Se puede citar como ejemplo el sitio MyYahoo, en donde se pueden encontrar allí muchos de los aspectos personalizados antes nombrados. En dicho sitio el usuario ingresa a una página en donde selecciona sus preferencias relacionadas a dichos aspectos.

En ciertos casos de personalización de colores, la aplicación ofrece esquemas armados con diferentes motivos que pueden seleccionar los usuarios, guardando en el perfil el estilo elegido por el mismo. De esta manera, al momento de armar la visualización se interactúa con el perfil obteniéndose el dato del motivo que seleccionó el usuario y que debe ir en el nodo. En otros casos, los colores de los componentes pueden ser seleccionados por separado, seleccionar colores de fondo, de título, entre otros. Con esta posibilidad el perfil de usuario debe ser capaz de almacenar la información que corresponda para luego, cuando se interactúe con él, se obtengan las preferencias del usuario relacionadas a los colores del nodo. Con respecto a las posiciones de los bloques de información, existe la posibilidad de poder ubicar los bloques de información de manera personalizada, en este caso el usuario decide dónde ubicar la información en relación a sus intereses, establece ver primero la información que más le interesa, y en la parte inferior del nodo ubicar la información menos importante, como se puede observar en el Figura 6.4. Aquí se debe almacenar en el perfil información sobre la ubicación de los bloques de información dentro del nodo, para cuando se interactúe por la visualización los bloques sean acomodados de la manera que el usuario escogió.

Figura 6.4: Cambio de posición del Bloque de información que contiene el Pronóstico.

Una vez que se resolvió toda la información referente a la visualización del nodo, se cumple con la última etapa de creación del nodo. Por lo tanto, el nodo está completo y es devuelto al Modelo de Navegación para que éste se encargue de devolverlo al Controlador, para que luego sea visualizado por el usuario.

6.4 Implementación de los Modelos y el Perfil de Usuario

En las secciones anteriores se presentó qué responsabilidades cumplen cada uno de los modelos que conforman el Modelo del MVC y el Perfil de Usuario para lograr la creación de los nodos navegacionales. Sin embargo, en dichas secciones no se expresó cómo es que ellos resuelven sus tareas. Este objetivo se desempeña en esta sección donde se presenta cómo debe ser realizada la implementación de la arquitectura propuesta.

La implementación del Modelo del MVC se encuentra completamente realizada utilizando programación orientada a objetos. Esta decisión se basa en primer lugar en los beneficios que dicha programación brinda. Se pueden nombrar su facilidad de desarrollo, mantenimiento, utilidad y escalabilidad, entre otros. Y en segundo lugar se tiene en cuenta en la decisión el auge que dicha programación tiene hoy en día y en la gran aceptación que posee en el mercado. Por dichas razones, todos los modelos y el perfil de usuario presentados pueden ser llevadas a cabo bajo la utilización de clases, métodos y conocimiento entre dichas clases. A continuación se describen características puntuales de cómo realizar la implementación de la arquitectura, además se exponen algunas decisiones que se tomaron sobre puntos importantes de la misma.

Un punto importante a destacar es la implementación del Modelo Navegacional que es donde comienza el armado de los nodos, y además, es el encargado de coordinar la comunicación con los demás modelos. Para comprender correctamente la implementación del mismo, se debe tener presente para esta explicación que los nodos son definidos en tiempo de diseño, y que dichos nodos, cuando la aplicación se encuentra ejecutándose, pueden ir cambiando en su creación cuando se van adaptando a la personalización. Por lo cual, existen clases nodos que contienen en sus métodos la manera de armar los nodos, e interactuando con las clases de los perfiles se obtiene la creación de los nodos de manera personalizadas. Para lograr la coordinación entre los tres modelos y el armado de los nodos se utiliza una clase abstracta Nodo. Dicha clase, contiene la implementación de un Template Method [54] cuya función es la de definir el esqueleto de un algoritmo a ejecutar, y que las subclases que heredan este método redefinan partes o el algoritmo completo sin cambiar su estructura. Observar la Figura 6.5 en donde se representa la estructura de las clases. Este template method define un algoritmo que contiene, primero un método que realiza un llamado a un método en una clase dentro del Modelo Navegacional que posee la responsabilidad de armar el nodo con los bloques de información, *obtenerBloques()*. Un segundo método que llama a un método en una clase del Modelo de la Aplicación para que realice el llenado del contenido del nodo, *obtenerContenido()*[19]. Y un tercer método, que llama a un método de una clase del Modelo de Interfaz encargada de completar los datos de interfaz, *obtenerInterfaz()*[1]. Existe una subclase nodo por nodo de la aplicación, esto sucede por el motivo de que los nodos necesitan diferentes elementos para ser armados y por lo cual deben interactuar con diferentes clases para lograr dicho objetivo. Entonces el template method será redefinido por cada una de las subclases. Contendrá información de cómo armar la estructura de los nodos y a qué clases conocer de los otros dos modelos para continuar con el armado de los mismos, respetando para realizar el armado la estructura del template method. Se basará en la estructura de este método para realizar el llamado a los métodos de los distintos modelos, sabiendo en cada método (obtenerBloques, obtenerContenido, obtenerInterfaz) con qué clase de los modelos le corresponde

19 En estos dos casos la interacción del template method será con una clase de los módulos, la cual luego, deberá realizar otras interacciones con clases de su modelo para lograr una respuesta personalizada y enviar dicha respuesta a la clase del Modelo Navegacional que comenzó la interacción.

interactuar para obtener la información necesaria para la creación de cada nodo. Una vez que se completa la ejecución de este template method, el nodo se encuentra completamente armado, y es devuelto al Controlador que fue el que llamó a dicho método para satisfacer el pedido realizado por el usuario.

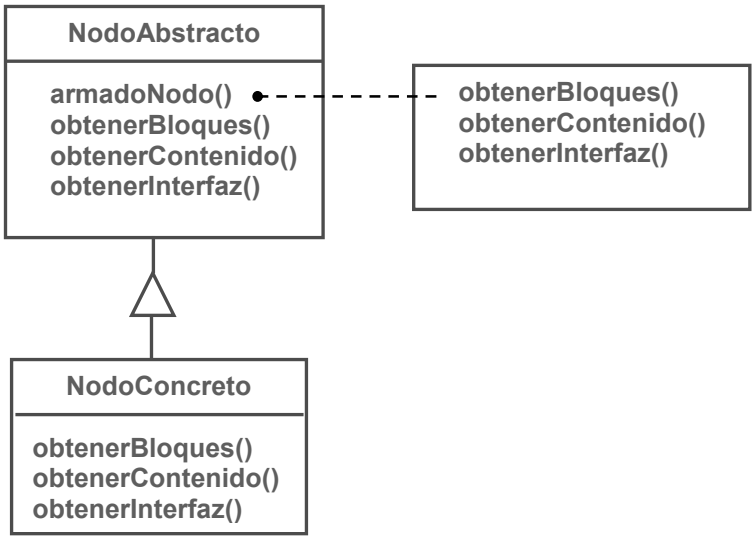

Figura 6.5: Uso del Template Method para el armado de los nodos.

La implementación del Modelo de Interfaz dependerá muchísimo de los aspectos de interfaz que se personalicen y de qué diferencias de visualización existan entre los nodos de la aplicación. Si se personaliza el aspecto de los colores, donde todos los nodos conservan la misma interfaz en cuanto a dicho aspecto, bastará solo con implementar una clase NodoInterfaz que posea las características de visualización implementando métodos que interactuando con el perfil de usuario puedan devolver la información necesaria. Por ejemplo, si sucede que el color de fondo, del texto y del título, se mantiene para todos los nodos igual, con una sola clase que implemente métodos para resolver los colores de dichos elementos interactuando con los perfiles de usuarios será suficiente para lograr el objetivo. En caso de que esto no ocurra de esta manera y puedan llegar a ser muy diferentes los aspectos y elementos personalizados entre los diferentes nodos, la solución se desarrollará de una manera muy similar a la utilizada para el Modelo de Navegación. En el sentido que se implementará una clase para cada nodo, la cual poseerá las características particulares que el nodo de la aplicación presente. Teniendo una superclase nodo de interfaz abstracta con características de interfaz generales de los nodos, y luego deben crearse subclases que redefinan los métodos de la superclase.

Con respecto a la implementación del Modelo de Aplicación sucede muy diferente a lo descripto para los otros dos modelos. Esto ocurre porque este modelo contiene la lógica de la aplicación, su dominio y sus reglas de negocio. Por lo cual, se implementarán las clases pertenecientes a representar los objetos del dominio, el conocimiento e interacción que entre ellos ocurre, y además, serán implementadas las clases para efectuar las reglas de negocio con todos los métodos que se necesiten para lograr desarrollar su lógica. Como se puede observar, este modelo no implementa ninguna clase que contenga aspectos de visualización cumpliendo de esta manera con la principal característica y objetivo de que este modelo no posea aspectos de interfaz, ni tenga ningún conocimiento de que ella existe. Por lo cual su implementación no refleja ninguno de los aspectos posibles a la visualización de los nodos.

Luego de haber presentado la implementación de cada uno de los modelos corresponde exponer cómo se realiza la personalización de los aspectos que le pertenece a cada uno de ellos. Para lograr la personalización de cada uno de sus aspectos, dentro de algunos métodos de las clases que se describieron para cada uno de modelos se deben realizar interacciones con el perfil del usuario para obtener sus preferencias y gustos. Para lograr esto, se debe interactuar primero con el manejador de perfiles de usuarios solicitando el perfil correspondiente al usuario que solicitó el pedido, y una vez que se obtiene dicho perfil, se debe realizar en cada método las acciones adecuadas para obtener del mismo la información necesaria para personalizar el aspecto del nodo.

Otro punto a destacar, es la existencia de la clase Perfil de Usuario. Esta clase se compone de información referente a las preferencias e intereses de los usuarios, almacenando dicha información en variables, y además por el conocimiento a una clase Usuario, que contiene los datos personales de cada uno de los usuarios, nombre, edad, localidad, nacionalidad, entre otras. Dependiendo de los aspectos de personalización que implemente la aplicación serán las variables necesarias que deben componer la clase perfil de Usuario y los métodos propios que dicha clase deba implementar para interactuar con las clases de los modelos para brindarles conocimiento de sus datos. Esta clase no tiene mucha lógica ya que es encargada solo de almacenar datos de los usuarios. También se implementará una clase con la responsabilidad de administrar todos los perfiles de usuarios que contenga la aplicación, dicha clase deberá poseer una colección que conservará todos los perfiles. Esta clase es la encargada de la creación, eliminación, actualización y modificación de los perfiles de usuarios, para lo cual deberá implementar métodos que cumplan con dichas operaciones. Asimismo, una de las tareas que debe desempeñar es la búsqueda del perfil de usuario que será pedido por alguno de los modelos, dado que los modelos interactúan con esta clase para pedir el perfil del usuario que necesiten para personalizar el nodo.

Con respecto al conocimiento que los modelos tienen del Perfil de Usuario se puede señalar porqué sucede de esta manera. Como se describió en las secciones anteriores cada uno de los modelos, en caso de personalizar aspectos que competen a sus responsabilidades, realizan interacciones con el perfil de usuario para obtener información acerca de las preferencias e intereses del mismo. Por dicha razón deben poseer acceso a los perfiles de usuarios, entonces cada modelo debe tener conocimiento del manejador de perfiles. Esto quizás podría evitarse siendo el Modelo Navegacional el que interactúe por primera vez con los perfiles y obtenga el perfil correspondiente del usuario, y de este modo, enviarlo de modelo a modelo sin la necesidad de que ellos deban ir a buscarlo también. Esta decisión de no realizarlo de esta manera se basó en la posibilidad de agregado o eliminación de aspectos de la personalización. En tal caso, si no es personalizado el aspecto de estructura el Modelo Navegacional no tendría la necesidad de conocer al manejador de perfiles, sin embargo sería obligatorio el conocimiento para que dicho modelo obtenga el perfil para luego enviarlo de modelo a modelo. Y además, no sería necesario quizás enviarlo de modelo a modelo dado que alguno de los modelos puede no poseer aspectos personalizados y en este caso no necesitaría del perfil. Para evitar estos dos inconvenientes, se tomó la decisión de que cada modelo interactúe con el manejador, en caso de así precisarlo, y le pedirá el perfil correspondiente al usuario que realizó el pedido, teniendo como ventaja que la operación de búsqueda del perfil es realizada en el servidor por lo tanto el tiempo de búsqueda es ínfimo.

Otro punto a desarrollar que se encuentra relacionado con los perfiles de los usuarios es cómo estos perfiles son actualizados con información que llega de los usuarios desde la vista en el cliente. Esta información viaja dentro de los parámetros junto con el pedido realizado por el usuario a la aplicación, llegando así al Controlador. Éste se comunica con el Modelo Navegacional enviándole, además de los pedidos realizados por los usuarios, los parámetros que contienen los datos sobre el comportamiento y preferencia del usuario. Debe existir alguien que sea el encargado de manejar filtrar y almacenar

dichos datos en los perfiles de usuarios correspondientes. Para realizar estas tareas existirá, además del manejador de perfiles, una clase dentro del modelo Perfil de Usuario que implemente métodos para realizar estas operaciones. A dicha clase el Modelo de Navegación le enviará los parámetros correspondientes a los datos del usuario y ella será la encargada de interactuar con el manejador de perfiles para obtener el perfil respectivo y con los datos que le llegaron actualizar y completar el perfil del usuario.

En esta sección se expuso la implementación de los puntos más importantes y críticos de la aplicación, dejando constancia de cómo deben ser llevados a cabo. Se desarrolló la explicación de la implementación de cada uno de los modelos del MVC junto con el Perfil de Usuario, y otros aspectos que puedan requerir de alguna explicación. De esta manera, se deja evidencia de que la realización de la arquitectura que fue descripta a lo largo de este trabajo puede ser llevada a cabo desarrollando para su implementación las clases que fueron presentadas en esta sección.

6.5 Conclusión

Se presentó en este capítulo el corazón de la arquitectura, el Modelo del MVC constituido por tres modelos y un Perfil de Usuario. Se expusieron cada una de las responsabilidades que cada modelo debe realizar, como se efectúa su interacción con los perfiles de los usuarios para lograr la personalización, y luego de presentado su funcionamiento, se describió cómo debe ser realizada la implementación de dichos modelos así como del Perfil de Usuario. Se expuso además alguna cuestión de implementación que era importante destacar.

Como se explicó anteriormente, los nodos de navegación se encuentran compuestos por bloques de información y, si bien estos nodos son diseñados de antemano, son creados y adaptados a la personalización dinámicamente en tiempo de ejecución de la aplicación cuando son requeridos por algún usuario. Para lograr la creación de un nodo, el mismo se va transfiriendo de modelo a modelo y cada uno de dichos modelos cumple con la tarea de completar una fracción del nodo. Para completar cada fracción del nodo cada modelo, según su responsabilidad, irá completando y adaptando la información que contendrá el bloque de información a las preferencias e intereses de cada usuario. Una vez que el nodo concluyó su pasaje a través de cada modelo, se obtiene como resultado el nodo completo y personalizado que es enviado de regreso al Controlador para que éste lo envíe al usuario que lo solicitó. De esta manera, la creación del nodo se va completando de a poco y cada uno de los modelos realizando su tarea aporta información para lograr armar al mismo, y además, cada uno de los modelos se encargará de personalizar el aspecto que es responsabilidad de él.

Con el desarrollo de tres modelos y el Perfil de Usuario se logra completar el objetivo principal de este trabajo que es el de realizar una arquitectura modular, teniendo en cuenta también para lograr este objetivo el primer nivel de la arquitectura. En este segundo nivel, el objetivo se consigue con la división de los modelos a partir de su comportamiento y funcionalidad, e integrando el Perfil de Usuario para obtener la personalización de la aplicación. Como se pudo observar en la descripción de estos modelos, existe independencia entre ellos en cuanto a diseño, implementación y mantenimiento de los mismos. La razón de que ello suceda se debe a que las responsabilidades que debe cumplir cada uno son totalmente diferentes e independientes una de otras, cumpliendo cada uno con una tarea para lograr la creación personalizada de los nodos de navegación. La idea de la división surge a partir de la separación de responsabilidad para obtener independencia entre los módulos y con ella los beneficios que aporta. Se pueden señalar como ventajas la posibilidad de la realización el diseño e implementación de manera separada basadas en las diferentes responsabilidades que debe cumplir cada modelo, logrando capacidad de extensión o modificaciones sobre la aplicación afectando con estos cambios a un sector mínimo de la aplicación, y evitando

de esta manera realizar cambios drásticos que pueden llegar a ser muy difíciles de llevar a cabo. Esta última ventaja quizás quedé más clara luego de la lectura del próximo capítulo. Otra ventaja que puede ser nombrada con respecto al Perfil de Usuario, es la ubicación elegida para el mismo. Contando por esta razón con la rapidez de respuesta en las interacciones que los modelos realizan con los perfiles, además de la posibilidad de utilización de la conocida técnica de personalización collaborative filtering.

Por último se debe resaltar, como se describe en la sección 6.4, cómo debe ser llevada a cabo la implementación de los modelos así como del Perfil de Usuario, y se puede observar que la implementación de cada modelo así como la del perfil de usuario pueden ser realizadas de manera totalmente independiente. Esto sucede por el motivo que los modelos no dependen uno de otros, sino más bien dependen de los aspectos de los nodos que deben crear cada uno de ellos y de los aspectos que pueden ser personalizados, logrando de esta manera la posibilidad de desarrollar una aplicación completamente modular, en donde se puede realizar el diseño, la implementación y el mantenimiento de una manera independiente entre los modelos que componen el Modelo de la arquitectura. Asimismo, cabe destacar como última ventaja que la implementación de los modelos y del Perfil de Usuario pueden ser llevada a cabo de manera sencilla utilizando programación orientada a objetos, un lenguaje muy conocido hoy en día en el mercado de las aplicaciones Web.

Capítulo 7

Funcionamiento de la Arquitectura

Este capítulo se encuentra dedicado a dar un cierre a este trabajo. Luego de haber expuesto en los dos capítulos anteriores una descripción de cada uno de los niveles que componen la arquitectura propuesta, se consideró adecuado dar una idea del funcionamiento completo de toda ella, expresando de manera más detallada como se realiza la interacción entre los dos niveles de la arquitectura. Además se describen como deben ser realizadas las modificaciones en cuanto al agregado o eliminación de aspectos de personalización, y cómo dichas modificaciones afectan a los módulos de la arquitectura.

Se compone solamente de dos secciones. La primera, sección 7.1, presenta una visión completa de la interacción que se produce entre los módulos de la arquitectura y el pasaje de información entre ellos para lograr armar el nodo personalizado solicitado por el usuario. La segunda y última sección, sección 7.2, presenta cómo deben ser realizados los cambios dentro de la arquitectura cuando se desean agregar o eliminar aspectos de la personalización. Se describe qué módulos se pueden ver afectados con dichos cambios, y se expone cómo estos cambios los afectan, demostrando la facilidad con la cual estos cambios pueden ser llevados a cabo.

7.1 Creación de los Nodos

En el capítulo 5 se presentó el primer nivel que compone a la arquitectura propuesta constituido por el patrón MVC. Luego, como un segundo nivel, en el capítulo siguiente se describió el componente Modelo de dicho patrón que forma parte de este primer nivel y que define el corazón de la arquitectura. Si bien fue explicado anteriormente parece adecuado presentar el funcionamiento completo de la arquitectura, dado que, en los estos capítulos se realizó la descripción de cada uno de los niveles pero sin poner mayor énfasis en la interacción completa desde que el usuario realiza el pedido hasta que la respuesta personalizada retorna al mismo. Por este motivo, se consideró adecuado presentar en esta sección la interacción completa de los componentes del MVC y de los módulos dentro del Modelo. Dicha interacción puede observarse en la Figura 7.1, en donde cada número indica la secuencia de pasos a seguir, y a continuación se describen cada uno de ellos.

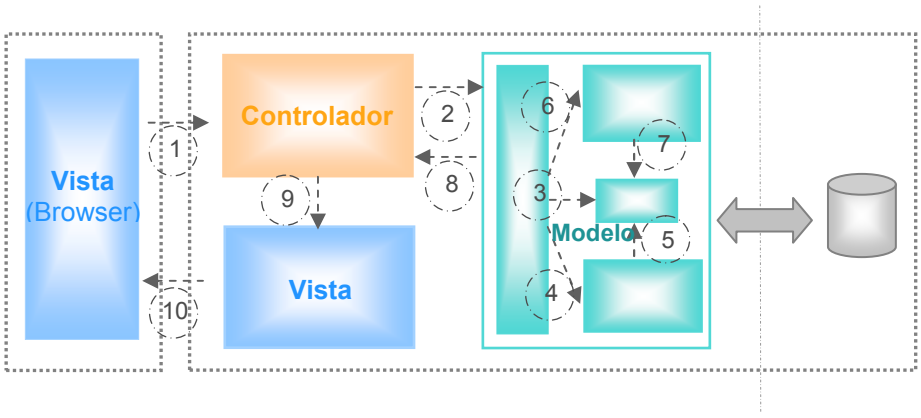

Figura 7.1: Interacción completa de los módulos de la arquitectura.

Como primer paso se puede observar que desde el browser se realiza un pedido a la aplicación *(1)*, que es derivado en un pedido realizado al componente Controlador. Dicho pedido viene acompañado de datos del usuario que realizó la petición, datos en el Modelo serán utilizados para la personalización. El Controlador traduce el pedido a una expresión entendible por el Modelo y realiza la solicitud del nodo que el usuario desea visualizar al Modelo Navegacional *(2)*. Una vez que se le realizó el pedido, el Modelo Navegacional interactúa con el Perfil de Usuario para armar la estructura del nodo de manera personalizada *(3)*, logrando un nodo con los bloques de información que lo componen. Con respecto a esta personalización en el Modelo Navegacional, puede ocurrir que para un mismo nodo la estructura pueda variar para dos usuarios distintos, dado que al ser personalizada algunos bloques de información pueden aparecer dentro del nodo que fue elegido por uno de los usuarios, pero para el otro usuario con diferente preferencia puede suceder que ese bloque no se encuentre dentro del nodo. Obsérvese Figura 7.2, en donde se exponen nodos diferentes que pueden ser creados por una misma clase Nodo.

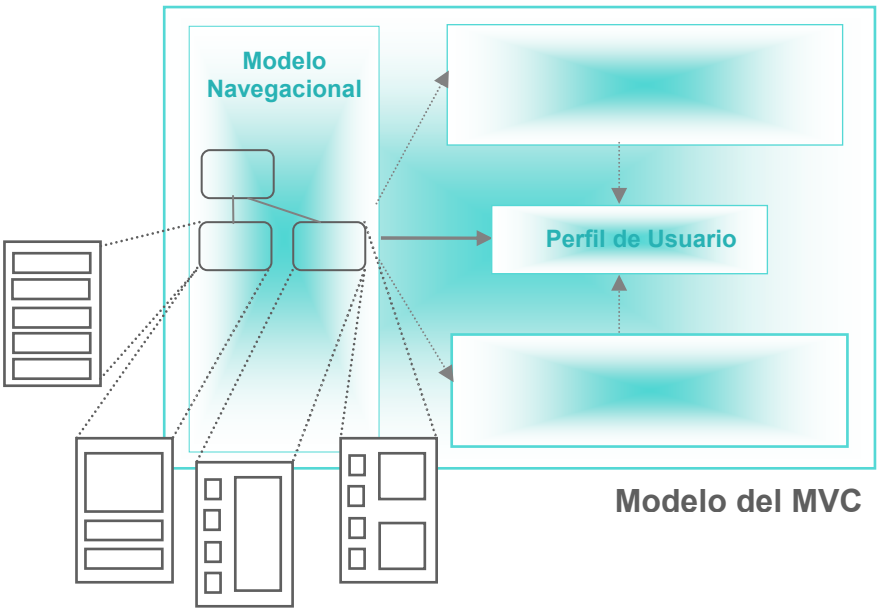

Figura 7.2: El Modelo Navegacional se encarga del armado de la estructura personalizada de los nodos.

Luego que se obtiene la estructura del nodo personalizado se efectúa un pedido al Modelo de Aplicación para obtener el contenido de información de cada uno de los bloques del nodo *(4)*. Este modelo realiza interacciones entre sus clases para lograr cumplir con su tarea utilizando, la lógica de la aplicación, los objetos del dominio, las reglas de negocio y el Perfil de Usuario *(5)*, obteniendo de este modo la información correcta y personalizada que debe ir en el nodo. Se puede observar en la Figura 7.3 como las clases llenan los mismos nodos con diferente información correspondiente a los diferentes usuarios.

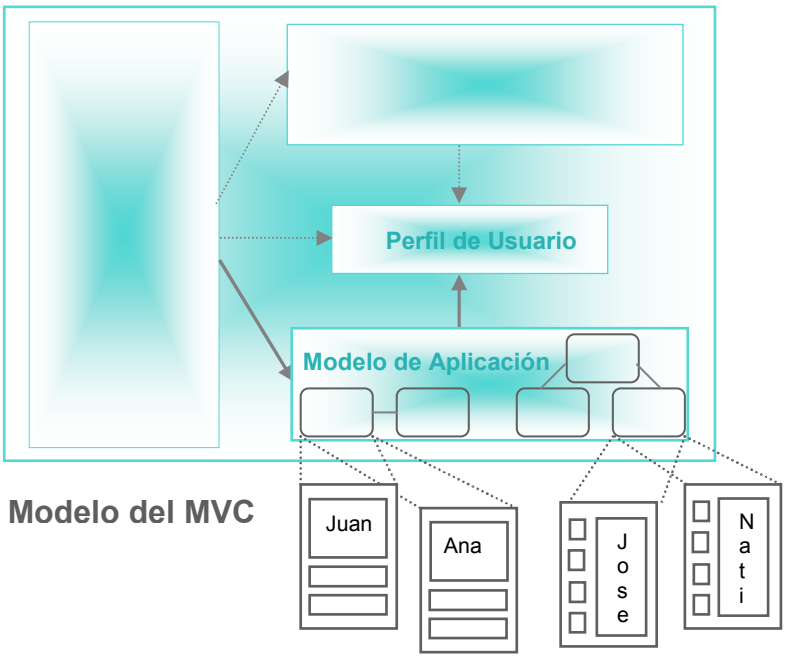

Figura 7.3: El Modelo de Aplicación se encarga del contenido de información personalizada de los nodos.

Cuando finaliza esta tarea retorna el nodo casi completo al Modelo Navegacional. Para terminar de completar la creación del nodo solo hacen falta las propiedades relacionadas con la visualización, y para lograr esta última etapa el Modelo de Navegación interactúa con el Modelo de Interfaz *(6)*. Éste recibe el pedido para completar el nodo e interactúa con sus clases y con el Perfil de Usuario *(7)* para lograr este objetivo. Y luego de terminada su tarea, retornando al Modelo Navegacional el nodo concluido. Se muestra en la Figura 7.4 como los nodos quedan finalizados agregando las propiedades para su visualización, se presentan dos nodos iguales con la misma estructura, que poseen diferentes colores y distinta posición en la que son ubicados los bloques de información basándose en las preferencias de los usuarios que solicitaron cada nodo.

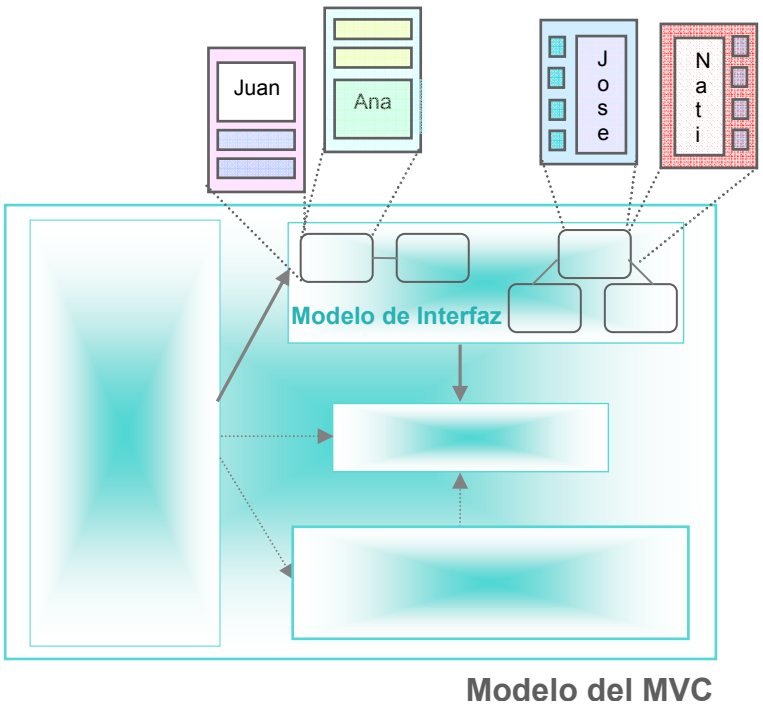

Figura 7.4: El Modelo de Interfaz se encarga de las propiedades personalizadas relacionadas a la visualización de los nodos.

Al finalizar la creación personalizada del nodo que fue pedido el Modelo Navegacional lo retorna al Controlador *(8)*. Ni bien lo recibe el Controlador envía el nodo completo a la Vista del lado del Servidor *(9)* para que ésta realice la tarea de transformar los datos del nodo en información entendible por la Vista que se encuentra del lado del Cliente. Una vez realizada esta trasformación, la información de resultado es enviada a la Vista del lado del cliente *(10)*, la cual es visualizada como un nodo personalizado en respuesta al pedido solicitado.

7.2 Facilidad Agregado y Eliminación de Aspectos personalizables

Esta sección se encuentra dedicada a exponer cómo deben ser realizados los cambios en cuanto a los aspectos de personalización de la aplicación dentro de la arquitectura propuesta. Qué módulos se ven afectados cuando se quiere agregar o eliminar algún aspecto, cómo impacta este cambio en el Perfil de Usuario, en el funcionamiento de la aplicación y sobre todo cual es la complejidad que la realización de estos cambios provoca.

Se destacó en el capítulo 5, donde se describe el primer nivel de la arquitectura, que este nivel se encuentra totalmente abstraído de los aspectos de personalización, y se señala que estos aspectos se encuentran encapsulados en el Modelo. Por dicha razón, los cambios y modificaciones que surgen como consecuencia de realizar cambios en los aspectos de personalización no afectan a los componentes de este nivel. Se puede destacar que dentro de este nivel los que pueden sufrir modificaciones son los parámetros que son enviados junto con el pedido realizado por la Vista del lado del Cliente al Controlador y luego al Modelo, que son los que transportan la información referente a los usuarios y quizás sea necesario enviar más información de ellos o menos, dependiendo de si se agrega o se elimina algún aspecto.

Al Modelo encapsular los aspectos de personalización de la aplicación los cambios que se realicen implican que serán afectados los módulos que componen este segundo nivel de la arquitectura. El primer punto importante a señalar, es que el agregado o eliminación de aspectos de personalización no sugieren una división diferente en cuanto a los módulos dentro Modelo del MVC, éste sigue compuesto por los tres modelos y el Perfil de Usuario, dado que las modificaciones que conciernen a la personalización de la aplicación se ve reflejada en cada modelo particular o en el perfil, pero no así se realizan modificaciones en la arquitectura propuesta, ésta se mantiene intacta. Como se describió en el capítulo anterior, cada módulo contiene una tarea que realiza y además respecto a la personalización, cada uno de ellos es encargado de personalizar ciertos aspectos. Por este motivo, cada módulo se verá afectado dependiendo del aspecto de personalización que se agregue o elimine. A continuación se describen módulo a módulo como se ven afectados dependiendo del aspecto de personalización a agregar o eliminar.

El Perfil de Usuario en general siempre va a ser afectado cuando se realicen cambios, dado que es quien contiene toda la información de los usuarios, y en caso de agregar uno o más aspectos a personalizar debe ampliarse la información que va a contener, además de agregarle manejo para disponer de ella. Pueden surgir casos en que se agreguen aspectos de personalización y que no deba ampliarse la información del mismo dado que ya la contiene. En el caso de eliminar algún aspecto, es opcional la eliminación de los datos del perfil que ya no son necesarios. Se deberá decidir si se desea limpiar el perfil, en este caso se elimina la información que ya no es precisa para la personalización y además se deben eliminar los métodos para el manejo de la misma, o en caso de no realizar una política de limpieza los datos quedan en el Perfil pero no serán utilizados. También irán cambiando los conocimientos que los modelos poseen de él. Si alguno de los modelos comienza a personalizar aspectos que son responsabilidad de él y antes no personalizaba ningún aspecto, dicho modelo pasará a poseer conocimiento de la existencia del Perfil de Usuario ya que tendrá que interactuar con él para personalizar los aspectos de los cuales está encargado. En caso contrario, que se elimine el aspecto que el modelo contenga para personalizar, el conocimiento con respecto al Perfil ya no existirá porque ya no será necesario realizar la interacción con el Perfil.

El Modelo Navegacional además de armar los nodos es el encargado de personalizar el aspecto de estructura de los nodos. Para armar los nodos se debe saber qué bloques de información contienen cada uno de ellos. En caso de no personalizar el aspecto de estructura, el Modelo Navegacional arma el nodo con los bloques que éste debe poseer, ejecutando un método para realizar dicha tarea. En caso de agregar este aspecto personalizado, dicho método es modificado para personalizar la estructura, por lo cual cuando se realice el armando del nodo el Modelo Navegacional deberá poseer conocimiento del Perfil de Usuario para interactuar con él, y de este modo, saber qué bloques de información serán incluidos en el nodo y cuales no según las preferencias del usuario. Si el Modelo Navegacional realizaba esta personalización y luego es eliminada, el conocimiento con el Perfil de Usuario será eliminado, el método de armado del nodo armará el nodo con los bloques de información por defecto para todos los usuarios igual. Con respecto a la responsabilidad de este modelo de interactuar con cada uno de los otros modelos, se puede destacar que no se ve afectada la secuencia en la que se produce

la interacción entre ellos, dado que como se señaló anteriormente, la arquitectura queda intacta en cuanto a su división de módulos cuando se realizan cambios en los aspectos a personalizar.

En cuanto al Modelo de Aplicación si no se realiza la personalización del aspecto de contenido, que es el aspecto del cual se encarga este modelo, no se posee conocimiento del Perfil de Usuario y se implementará como el Modelo de una aplicación común. En cambio, cuando se agrega la personalización en este modelo es un poco más difícil destacar puntualmente cuales van a ser las modificaciones para poder realizar la implementación de este cambio. Sucede con este modelo que puede ser personalizada diferente información dentro de los bloques de información de cada nodo, para los cuales se obtiene su contenido interactuando con diferentes clases, y además quizás con reglas de negocio para obtener la información que debería ir en el bloque. Por ello, las modificaciones que deben ser realizadas para este aspecto pueden ser no solo el agregado o eliminación de este aspecto en general, sino el agregado de personalización en cuanto al contenido de un bloque en particular. Por dicha razón, se debe modificar la clase que contiene la lógica para resolver el contenido del bloque con la información a personalizar, la cual debe interactuar con otras clases para resolverlo y con el Perfil de Usuario. Esta clase modificada puede ser tanto una clase de lógica del dominio de la aplicación o una clase que ejecuta una regla de negocio. Para sintetizar un poco lo dicho, los cambios de este aspecto afectarán a las clases encargadas de resolver el contenido del bloque que debe ser personalizado interactuando con el Perfil de Usuario. En caso de eliminarse la personalización del contenido de un bloque de información, se debe hacer ingeniería inversa, se debe eliminar de la clase encargada de resolver esa información la implementación que compete a la interacción con el Perfil de Usuario, dejándola como una clase que resuelve un contenido sin personalizar. Si se quiere eliminar la personalización del aspecto de contenido, todas las clases que se encuentran afectadas deben eliminar su implementación de personalización y además se eliminará del Modelo de Aplicación el conocimiento con el Perfil de Usuario. Igualmente, uno de los aspectos más utilizados en el mercado de comercio electrónico es la personalización del aspecto de contenido, principalmente por la realización de recomendaciones de productos, muy común en estas aplicaciones, y por el reconocimiento en cuanto a darle la bienvenida a cada uno de los usuarios. Por estas dos razones es muy probable que este aspecto se encuentre presente cuando se realizase una aplicación Web personalizada.

Por último quedan por describir los cambios que deben ser realizados en el Modelo de Interfaz para adecuar la visualización de los nodos de cada usuario a la personalización. Este modelo es el encargado de personalizar los aspectos de interfaz que tienen que ver con la visualización de los bloques de información en cuanto a colores, formato de los textos (fuente, tamaño, entre otros), y posiciones de los bloques de información dentro de los nodos. Si la aplicación no personaliza ninguna de las propiedades antes nombradas, este modelo no posee conocimiento del Perfil de Usuario, y además obtiene todas las características de la interfaz en la clase que le corresponde resolver las particularidades visuales del nodo, puede ser una sola clase en caso que todos los nodos se visualicen con las mismas características, o una clase por cada nodo en caso de que sean visualizados diferentes. De una manera similar sucede cuando se agrega la personalización de dichas propiedades, si todos los nodos conservan la misma interfaz, bastará modificar la clase que resuelve las características visuales de todos los nodos modificando sus métodos para que realicen interacciones con el Perfil de Usuario. En caso que los nodos se visualicen de diferente manera se tendrá una clase nodo para cada nodo de la aplicación con las características del mismo. Dentro de la clase de cada nodo, se implementarán métodos que resuelvan la visualización del mismo interactuando con el Perfil de Usuario. Obteniendo de esta manera para dicho nodo, los colores, el formato de los textos, y si también se realiza la personalización de las posiciones de los bloques, la ubicación de cada bloque. En caso de poseer solo una de estas propiedades el método interactuará con el Perfil para obtener solo la información necesaria para esa

personalización, en otro caso, necesitará pedirle más información y así lograr toda la personalización. En este modelo las modificaciones a realizar quedan centralizadas en la clase que resuelven la visualización del nodo que fue requerido realizando interacciones con el Perfil de Usuario para obtener la información necesaria para lograr dicha tarea. Cuando se quiera eliminar los aspectos de personalización de este modelo, solo se deben modificar las clases que resuelven la visualización de los nodos dejando de interactuar con el Perfil de Usuario, obteniendo de este modo la misma información para todos los usuarios.

Se puede observar que al encontrarse encapsulados los distintos aspectos que pueden ser personalizados en los diferentes modelos, el agregado o la eliminación de ellos dentro de los mismos pueden ser llevados a cabo de manera separada e independiente, sin que el agregado de un aspecto a un modelo afecta a alguno de los otros dos.

Capítulo 8

Conclusiones

Como primer paso de este trabajo se investigaron y luego se presentaron las características y peculiaridades que poseen las aplicaciones personalizadas, obteniendo de esta manera una perspectiva de las funcionalidades que deberían ser soportadas por la arquitectura a desarrollar. Una vez que se concluyó con dicha investigación se comenzó con la búsqueda y examinación de metodologías de diseño, patrones de diseño, tecnologías utilizadas para estas aplicaciones y otras arquitecturas existentes que podrían aportar algún conocimiento y colaboraran con el trabajo de desarrollo que debía lograrse. De esta investigación surgieron las contribuciones que aportaron el patrón MVC, la metodología OOHDM y los patrones de personalización Web, y por último se deben destacar las arquitecturas expuestas en el capítulo3 que ampliaron la perspectiva de las implementaciones existentes para soportar aplicaciones personalizadas.

El aporte principal de este trabajo fue el desarrollo de una arquitectura que soporte la realización de aplicaciones Web personalizadas de gran envergadura, teniendo en cuenta todos los aspectos que pueden ser personalizados y la manera de resolver cada uno de ellos. Se desarrolló una arquitectura que se encuentra compuesta por varios módulos los cuales cumplen un rol fundamental y poseen diferente funcionalidad. El propósito logrado con dicha división es el de brindar flexibilidad, facilidad y escalabilidad en cuanto al diseño, la implementación y el mantenimiento de una aplicación personalizada compleja. La arquitectura propuesta se compone de dos niveles, el primer nivel constituye la comunicación de los usuarios con la aplicación, utilizando para su desarrollo el patrón MVC, y el segundo nivel que compone el corazón de la aplicación, encapsula los aspectos de la personalización y es el encargado de crear los nodos personalizados en respuesta a los pedidos realizados por los usuarios. Cada uno de estos niveles se encuentra dividido en módulos, obteniendo de esta manera, una arquitectura modular basando su división en diferentes funcionalidades que cumple cada uno de estos ellos. Logrando la independencia deseada para conseguir cumplir con el objetivo de realizar el diseño, implementación y mantenimiento de los módulos de manera separada, y con dicha independencia, lograr que los cambios a efectuar afecten sectores determinados sin propagar modificaciones al resto de los módulos, tratando de este modo, de centralizar los cambios a realizar a cada módulo particular. Con respecto al segundo nivel de la arquitectura, se puede diseñar un módulo completo sin tener necesidad de ir diseñando los otros al mismo tiempo, es verdad que se encuentran muy relacionados entre ellos pero pueden ser diseñados más tarde. Además se encuentra determinado qué módulo se encarga de qué aspectos de la personalización, por lo cual, un módulo puede estar incluyendo personalización y los otros no, y los aspectos que afectan a un módulo no se encuentran vinculados a los aspectos que afectan a los demás. Por este motivo, se puede lograr que las modificaciones en cuanto al agregado o eliminación de aspectos de personalización no propaguen modificaciones en los módulos que no son encargados de dicho aspecto y de esta forma encapsular las modificaciones a algunos sectores.

Se puede destacar que esta arquitectura se encuentra orientada a mejorar la performance y además a no delegar responsabilidades en la máquina del usuario, esto sucede dado que los perfiles de usuarios y la lógica para su manejo se encuentran almacenados en el servidor, logrando que las interacciones con los perfiles de los usuarios para lograr la personalización de los nodos se pueda llevar a cabo con mayor

rapidez. Se puede señalar una desventaja con respecto a esta decisión, los perfiles de usuario son solo utilizados por la aplicación desarrollada y otras aplicaciones no tienen acceso a los mismos y no pueden reutilizarlos para sus beneficios, por ejemplo personalización basada en estos perfiles ya recolectados.

También se presenta en este trabajo la implementación de la arquitectura, probando de esta manera que es posible desarrollarla con las tecnologías utilizadas hoy en día para las aplicaciones Web. Para el primer nivel se expuso una posible implementación utilizando el framework Struts, pero solo como posible ya que puede ser implementada utilizando otras tecnologías existentes. No obstante, para el segundo nivel se expuso una implementación un poco más acotada en el sentido que se utiliza programación orientada a objetos. El motivo que ocasiono esta elección fue la potencia, robustez y facilidad de desarrollo que proporciona esta programación para expresar el armado de los nodos utilizando patrones y clases.

Las aplicaciones Web personalizadas van a continuar sufriendo avances en cuanto a los posibles aspectos a personalizar acompañadas del avance tecnológico que sufre nuestra ciencia día a día. Se puede esperar que la arquitectura presentada en este trabajo posea la capacidad de soportar nuevos requerimientos de aspectos personalizables, y pueda ser extendida fácilmente, o adaptada a los cambios venideros.

Tabla de Figuras

Figura 1.1: Selección de preferencias para el perfil de búsqueda. 12
Figura 1.2: Diferentes páginas dependiendo de cada usuario. 13
Figura 1.3: Uso del My en los sitios personalizados, My eBay.com. 15
Figura 2.1: Ingreso a MyYahoo con Usuario y Contraseña 23
Figura 2.2: El usuario es reconocido automáticamente por la aplicación 24
Figura 2.3: El Perfil de Usuario figurativamente se compone de dos secciones. 27
Figura 3.1: Arquitectura que utiliza un mediator para obtener personalización. 35
Figura 3.2: Interacción entre las partes de la Arquitectura. 36
Figura 3.3: Framework de Personalización utilizando Smart Cards 38
Figura 3.4: Agente almacenado en el Proxy. 39
Figura 3.5: Las tres capas que componen la arquitectura. 41
Figura 4.1: Modelo del patrón MVC. 48
Figura 4.2: Link Personalizado en Amazon 50
Figura 4.3: Página de recomendaciones de Amazon. 50
Figura 4.4: Selección de estructura Personalizada en MyYahoo. 52
Figura 4.5: Estructura Personalizada en MyYahoo. 52
Figura 4.6: Yahoo como servicio de búsqueda CNN. 53
Figura 4.7: Búsqueda en Amazon, brindando el servicio. 54
Figura 4.8: Búsqueda dentro de Amazon. 54
Figura 5.1: Patrón MVC en la arquitectura. 60
Figura 5.2: Comunicación entre los componentes del MVC. 61
Figura 5.3: Existen diferentes tipos de Interfaz. 64
Figura 5.4: Diagrama de secuencia de una Petición. 65
Figura 5.5: Interacción en el framework Struts. 66
Figura 6.1: Modelo interno de la arquitectura. 70
Figura 6.2.a: Almacenamiento en el cliente y procesamiento en el servidor 74
Figura 6.2.b: Almacenamiento y procesamiento en el servidor. 74
Figura 6.2.c: Almacenamiento y procesamiento en el servidor. 74
Figura 6.3: Bloques de información que contiene el nodo. 75
Figura 6.4: Cambio de posición del Bloque de información que contiene el Pronóstico. 80
Figura 6.5: Uso del Template Method para el armado de los nodos. 82
Figura 7.1: Interacción completa de los módulos de la arquitectura. 87
Figura 7.2: El Modelo Navegacional se encarga del armado de la estructura personalizada de los nodos. 88
Figura 7.3: El Modelo de Aplicación se encarga del contenido de información personalizada de los nodos. 88
Figura 7.4: El Modelo de Interfaz se encarga de las propiedades personalizadas relacionadas a la visualización de los nodos. 89

Referencias

[1] *"Personalization Consortium"*.
http://www.personalization.org/

[2] Barry Leiner, Vinton Cerf, David Clark, Robert Kahn, Leonard Kleinrock, Daniel Lynch, Jon Postel, Lawrence Roberts, Stephen Wolff, *"Una breve historia de Internet"*.
http://www.ati.es/DOCS/internet/histint/histint1.html

[3] Michael J. Pazzani, *"Commercial Applications of Machine Learning for Personalized Wireless Portals"*.
www.ics.uci.edu/~pazzani/Publications/Pricai2002.pdf

[4] Laurent Frelechoux y Tomonari Kamba, *"An architecture to support personalized Web applications"*.
http://www.ra.ethz.ch/CDstore/www6/Posters/726/POSTER726.html

[5] Mike Perkowitz y Oren Etzioni, "*Adaptive Web Sites"*, Communications of the ACM, pág. 152-158, volumen 43, número 8, Agosto 2000.

[6] Gustavo Rossi, Daniel Schwabe, Fernando Lyardet, *"Patterns for Designing Navigable Information Spaces*". Pattern Languages of Programs IV, Addisson Wesley, 1999.

[7] Nigel Wells y Jeff Wolfers, "*Finance with a Personalized Touch"*, Communications of the ACM, pág.31 - 34, volumen 43, número 8, Agosto 2000.

[8] Phillip J. Windley, "*Content Management, Portals, and Personalization"*, Julio 2002.
http://www.das.state.ut.us/cc/jul2002/ContentManagement.pdf

[9] Monica Bonett, "*Personalization of Web Services: Opportunities and Challenges"*.
http://www.ariadne.ac.uk/issue28/personalization/

[10] Alfred Kobsa, *"Personalized Hypermedia and International Privacy"*, Communications of the ACM, pág. 64-67, volumen 45, número 5, 2002.

[11] Lorrie Faith Cranor, *" 'I Didn't Buy it for Myself' Privacy and Ecommerce Personalization"*. ACM workshop on Privacy in the electronic society, pág. 111-117, 2003.

[12] Eugene Volokh, *"Personalization and Privacy"*, Communications of the ACM (Association for Computing Machinery), volumen 43, número 8, Agosto 2000.

[13] Cynthia A. Thompson, Mehmet H. Göker, Pat Langley, *"A Personalized System for Conversational Recommendations"*, JAIR (Journal of Artificial Intelligence Research) pág. 393-428, volumen 21, 2004.

[14] Sam Nelson "Dynamic, Static, and Generated Pages", 2005.
http://sps.clevernamehere.com/sps/manual/whitepapers/pagetypes.html

[15] Udi Manber, Ash Patel y John Robison, *"Experience with Personalization on Yahoo!",* Communications of the ACM, pág. 35-39, volumen 43, número 8, Agosto 2000.

[16] James Mendelsohn, *"BigAdmin System Administration Portal"*, 16 de Mayo 2001.
http://www.sun.com/bigadmin/content/developer/howtos/caching_part1.html

[17] Stephen Downes, *"Authentication and Identification"*, Mayo 2005.
http://www.downes.ca/cgi-bin/page.cgi?post=12

[18] Marek Czarkowski, Judy Kay*, "How to give the user a sense of control over the personalization of AH?"*, 2006.

[19] Magdalini Eirinaki y Michalis Vazirgiannis, *"Web Mining for Web Personalization"*.
http://www.db-net.aueb.gr/magda/papers/TOIT-webmining_survey.pdf

[20] Matteo Baldoni, Cristina Baroglio, and Nicola Henze, *"Personalization for the Semantic Web"*, Reasoning Web, First International Summer School 2005, Msida, Malta, Julio, 2005.
http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2005/personalization4SW_v02.pdf

[21] Jcorporate Ltd *"Personalization"*, 2001.
http://www.jcorporate.com/html/products/platform/personalization.html

[22] Liana Razmerita, Albert Angehrn, and Alexander Maedche, *"Ontology-based User Modeling for Knowledge Management Systems"*, 9th International Conference on User Modeling, Pittsburgh, USA, Springer-Verlag, pág. 213-217.
http://www.calt.insead.edu/Project/OntoLogging/documents/2003-UM-Ontology_based_user_modeling_for_Knowledge_Management_Systems.pdf

[23] Barry Smith y Paul Cotter, *"A Personalized Television Listing Service",* Communications of the ACM, pág. 35-39, volumen 43, número 8, Agosto 2000.

[24] A. Arsanjani, *"Analysis, Design, and Implementation of Dstributed Java Business Frameworks Using Domain Patterns"*. Tecnología de lenguajes

Orientado a Objetos y Sistemas 30. IEEE Computer Society Press, pág. 490-500, 1999.

[25] Gustavo Rossi, Andres Fortier, Juan Cappi, *"Seamless Personalization of E-Commerce Aplication"*. Workshop on Conceptual Modeling in E-Commerce, Japón, Diciembre 2001.

[26] Tata, consultancy services, *"Infrex, where business rules"*. http://www.tcs.com/0_products/infrex/

[27] Susanne Boll, *"Modular Content Personalization Service Architecture for E-Commerce Applications"*. 4th IEEE Int'l Workshop on Advanced Issues of E-Commerce and Web-Based InformationSystem (WECWIS 2002).

[28] Olivier Potonniée, *"Ubiquitous Personalization: A Smart Card Based Approach"*. http://www.gemplus.com/smart/rd/publications/pdf/Pot02per.pdf

[29] Laurent Frelechoux y Tomonari Kamba, *"An architecture to support personalized Web applications"*. http://www.ra.ethz.ch/CDstore/www6/Posters/726/POSTER726.html

[30] Chatree Sangpachatanaruk y Taieb Znati, "*A P2P Overlay Architecture for Personalized Resource Discovery, Access, and Sharing over the Internet"*. www2.sis.pitt.edu/~chatrees/Papers/ccnc05.pdf

[31] Donna Wolff, *"peer-to-peer"*, Febrero 2004. http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci212769,00.html

[32] Daniel Schwabe y Gustavo Rossi. *"An Object Oriented Approach to Web-Based Application Design"*. Theroy and Practice of Object Systems (TAPOS). Special Issue on the Internet, pág. 207-225, volumen 4, número 4, Octubre 1998.

[33] E. Gamma, R. Helm, R. Johnson, J. Vissides, *"Design Patterns. Elements of reusable object-oriented software"*, Addison Wesley , pág. 4-6, 1995.

[34] G. Rossi, D. Schwabe, J. Danculovic y L. Miaton, *"Patterns for personalized web applications"*, presentado en EuroPLoP 2001, Irsee, Alemania. http://hillside.net/patterns/EuroPLoP2001/

[35] Gustavo Rossi, *"An Object-Oriented Method for Designing Hypermedia Applications"*. PHD Tesis, Departamento de Informática, PUC-Rio, Brasil, Julio 1996 (en Portugués).

[36] Sun Microsystems, *"Object-Oriented Programming Concepts"*. http://java.sun.com/docs/books/tutorial/java/concepts/

[37] David Lowe, Brian Hernderson-Sellers y Alice Gu, "*Web extensions to UML: Using the MVC Triad*". Lecture Notes In Computer Science, pág. 105-119, volumen 2503. Proceedings of the 21st International Conference on Conceptual, 2002.

[38] D. Cowan and C. Lucena. Abstract Data Views, *"An Interface Specification Concept to Enhance Design for Reuse"*. IEEE Transactions on Software Engineering. volumen 21, número 3, Marzo 1995.

[39] Colin Moock, "*Essential ActionScript 2.0, Object-Oriented Development with ActionScript 2.0"*. O'Reilly Media, Capítulo 18, pág. 386-422, 2004.

[40] Wikipedia, La enciclopedia libre, *"Modelo Vista Controlador"*.
http://es.wikipedia.org/wiki/Modelo_Vista_Controlador.

[41] Malcolm Davis, *"Struts, an open-source MVC implementation"*, Febrero 2001.
http://www-128.ibm.com/developerworks/web/library/j-struts/

[42] Sun One Architecture Guide, *"Presentation Frameworks, Overview of Presentation Frameworks"*. Capítulo 8, pág. 121-125.
http://www.sun.com/software/sunone/docs/arch/

[43] Brian Kotek, *"MVC design pattern brings about better organization and code reuse"*, Octubre 2002.
http://builder.com.com/5100-6386-1049862.html

[44] World Wide Web Consortium (W3C).
http://www.w3.org/

[45] Sun Microsystems, *"Java BluePrints, Model-View-Controller"*.
http://java.sun.com/blueprints/patterns/MVC-detailed.html

[46] The Apache Software Foundation, *"Struts"*, 2000-2005.
http://struts.apache.org/userGuide/index.html

[47] Hans Bergsten, *"An Introduction to Java Servlets"* .
http://www.webdevelopersjournal.com/articles/intro_to_servlets.html

[48] JSPOLYMPUS, "Introduction to JSP".
http://www.jspolympus.com/JSP/JSP.jsp

[49] E. Gamma, R. Helm, R. Johnson, J. Vissides, *"Design Patterns. Elements of reusable object-oriented software"*, Addison Wesley, pág. 233-242, 1995.

[50] Sun Developer Network, "Java Server Pages [TM] techonoly", 2005.
http://java.sun.com/products/jsp/whitepaper.html

[51] By Chuck Cavaness, *"Using Tiles*, *Programming Jakarta Struts"*, O' Reilly, Cap. 14, Noviembre 2002.

[52] Scott Kirsner, *" Close Encounters"*. CIO Magazine - Octubre 1997.
http://www.cio.com/archive/webbusiness/100197_main.html

[53] Gustavo Rossi, Andrés Fortier, Juan Cappi, *"Mapping Personalization Policies into Software Structures*". International Workshop on Recommendation and Personalization in E-Commerece, Malaga, España, Mayo 2002.

[54] E. Gamma, R. Helm, R. Johnson, J. Vissides, *"Design Patterns. Elements of reusable object-oriented software"*, Addison Wesley, pág 325-330, 1995.